

Matthias Klusch
Matthias Thimm
Marcin Paprzycki (Eds.)

LNAI 8076

Multiagent System Technologies

11th German Conference, MATES 2013
Koblenz, Germany, September 2013
Proceedings

 Springer

Lecture Notes in Artificial Intelligence 8076

Subseries of Lecture Notes in Computer Science

LNAI Series Editors

Randy Goebel

University of Alberta, Edmonton, Canada

Yuzuru Tanaka

Hokkaido University, Sapporo, Japan

Wolfgang Wahlster

DFKI and Saarland University, Saarbrücken, Germany

LNAI Founding Series Editor

Joerg Siekmann

DFKI and Saarland University, Saarbrücken, Germany

Matthias Klusch Matthias Thimm
Marcin Paprzycki (Eds.)

Multiagent System Technologies

11th German Conference, MATES 2013
Koblenz, Germany, September 16-20, 2013
Proceedings



Springer

Volume Editors

Matthias Klusch

German Research Center for Artificial Intelligence (DFKI) GmbH

Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany

E-mail: klusch@dfki.de

Matthias Thimm

University of Koblenz-Landau

Institute for Web Science and Technologies (WeST)

Universitätsstr. 1, 56070, Koblenz, Germany

E-mail: thimm@uni-koblenz.de

Marcin Paprzycki

Polish Academy of Sciences

Systems Research Institute

ul. Newelska 6, 01-447 Warsaw, Poland

E-mail: marcin.paprzycki@ibspan.waw.pl

ISSN 0302-9743

e-ISSN 1611-3349

ISBN 978-3-642-40775-8

e-ISBN 978-3-642-40776-5

DOI 10.1007/978-3-642-40776-5

Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013946848

CR Subject Classification (1998): I.2.11, I.2.8, K.4.4, H.3.4, H.4.1, I.6.5, I.6.8, I.2.3-4

LNCS Sublibrary: SL 7 – Artificial Intelligence

© Springer-Verlag Berlin Heidelberg 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

These are the proceedings of the 11th German conference on Multiagent System Technologies, which was held during September 16 - 20, 2013, in Koblenz, Germany. This year's edition of the MATES series was accompanied by a doctoral mentoring and one workshop dedicated to different issues of agent technology. The workshop was realized in cooperation with the fourth realization of JAWS (Joint Agent-Oriented Workshops in Synergy) and as such became part of the European joint event MATES+JAWS on agent technology. The MATES 2013 conference was organized in cooperation with the Distributed Artificial Intelligence chapter of the German Society for Informatics (GI), and sponsored by the GI. Moreover, it was co-located with the 36th German Conference on Artificial Intelligence and the 43th Symposium of the German Society for Informatics (Informatik 2013).

The set of regular MATES 2013 conference talks covered a broad area of topics of interest ranging from issues of agent-based coordination to simulation to negotiation. In keeping with its tradition, MATES 2013 also offered three invited keynotes on relevant topics in the broad area of intelligent agent technology, which were given by well-known, reputed scientists in the domain, and, in this year, were shared with KI 2013. Furthermore, the MATES doctoral mentoring program supported PhD students in their research and application of multiagent system technologies by providing the opportunity to present and intensively discuss their work with other students and experts of the field.

As Co-chairs and on behalf of the MATES Steering Committee, we are very thankful to the authors and invited speakers for contributing to this conference, the Program Committee members and additional reviewers for their timely and helpful reviews of the submissions, as well as the local organizing team at the University of Koblenz, especially Ruth Ehrenstein, for helping to make MATES 2013 a success. Besides, we are indebted to Alfred Hofmann and the whole Springer LNAI team for their very kind and excellent support in publishing these proceedings.

Finally, we hope you enjoyed MATES 2013 and got some inspiration and insights from this technology domain that were useful for your own work!

July 2013

Matthias Klusch
Marcin Paprzycki
Matthias Thimm

Organization

Conference Chairs

Matthias Klusch	German Research Center for Artificial Intelligence, Germany
Marcin Paprzycki	Systems Research Institute of the Polish Academy of Sciences, Poland
Matthias Thimm	Institute for Web Science and Technologies, University of Koblenz-Landau, Germany

Doctoral Mentoring Chair

Lars Braubach	University of Hamburg, Germany
---------------	--------------------------------

Workshop Chairs

Costin Badica	University of Craiova, Romania
Christian Derksen	University Duisburg-Essen, Germany
Giancarlo Fortino	University of Calabria, Italy
Maria Ganzha	University of Gdansk, Poland
Ryszard Kowalczyk	Swinburne University of Technology, Australia
Marcin Paprzycki	Systems Research Institute Polish Academy of Sciences, Poland
Shahram Rahimi	Southern Illinois University, USA

MATES Steering Committee

Matthias Klusch	DFKI, Germany
Winfried Lamersdorf	University of Hamburg, Germany
Jörg P. Müller	Clausthal University of Technology, Germany
Paolo Petta	University of Vienna, Austria
Ingo Timm	University of Trier, Germany
Rainer Unland	University of Duisburg-Essen, Germany

Program Committee

Klaus-Dieter Althoff	DFKI, Germany
Vicent Botti	Universitat Politècnica de València, Spain
Lars Braubach	University of Hamburg, Germany

VIII Organization

Paul Davidsson	Malmö University, Sweden
Yves Roland Demazeau	CNRS, France
Jörg Denzinger	University of Calgary, Canada
Frank Dignum	Universiteit Utrecht, The Netherlands
Jürgen Dix	Technische Universität Clausthal, Germany
Torsten Eymann	University of Bayreuth, Germany
Maria Ganzha	System Research Institute Polish Academy of Sciences, Poland
Paolo Giorgini	University of Trent, Italy
Marie-Pierre Gleizes	University of Toulouse, France
Vladimir Gorodetsky	St. Petersburg Institute for Informatics and Automation, Russia
Verena Hafner	Humboldt University of Berlin, Germany
Koen Hindriks	Delft University of Technology, The Netherlands
Michael N. Huhns	University of South Carolina, USA
Wolfgang Ketter	Erasmus University, The Netherlands
Stefan Kirn	University of Hohenheim, Germany
Franziska Klg	Örebro University, Sweden
Ryszard Kowalczyk	Swinburne University of Technology, Australia
Daniel Kudenko	University of York, UK
Winfried Lamersdorf	University of Hamburg, Germany
Beatriz Lopez	University of Girona, Spain
Jörg Müller	Technische Universität Clausthal, Germany
Eugenio da Costa Oliveira	University of Porto, Portugal
Andrea Omicini	Università di Bologna, Italy
Sascha Ossowski	University Rey Juan Carlos, Spain
Mathias Petsch	Technische Universität Ilmenau, Germany
Paolo Petta	Universität Wien, Austria
Alexander Pokahr	University of Hamburg, Germany
Wolfgang Renz	HAW Hamburg, Germany
Rene Schumann	University of Applied Sciences West-Switzerland, Switzerland
David Sislak	Czech Technical University in Prague, Czech Republic
Ingo Timm	University of Trier, Germany
Adeline Uhrmacher	University of Rostock, Germany
Rainer Unland	University of Duisburg-Essen, Germany
Gerhard Weiss	Maastricht University, The Netherlands
Laszlo Zsolt Varga	Eötvös Lorand University, Hungary

Table of Contents

Invited Keynote

The Headaches of a Negotiation Support Agent	1
<i>Catholijn Jonker</i>	
Autonomous Systems Inspired by Biology	2
<i>Gerhard Weiß</i>	
AI-Research and the Future of the Automobil	3
<i>Raúl Rojas</i>	

Main Track

Similarity-Based Resource Retrieval in Multi-agent Systems by Using Locality-Sensitive Hash Functions	4
<i>Malte Aschermann and Jörg P. Müller</i>	
A Multi-agent Flooding Algorithm for Search and Rescue Operations in Unknown Terrain	19
<i>Matthias Becker, Florian Blatt, and Helena Szczerbicka</i>	
A Practical Security Infrastructure for Distributed Agent Applications	29
<i>Lars Braubach, Kai Jander, and Alexander Pokahr</i>	
Verifying MSMAS Model Using SCIFF	44
<i>Emad Eldeen Elakehal, Marco Montali, and Julian Padget</i>	
Decentralised Cooperative Agent-Based Clustering in Intelligent Traffic Clouds	59
<i>Jelena Fiosina, Maksims Fiosins, and Jörg P. Müller</i>	
Evolving Mechanisms in Boolean Games	73
<i>Cristiano Galafassi and Ana L.C. Bazzan</i>	
Repeated Sequential Single-Cluster Auctions with Dynamic Tasks for Multi-Robot Task Allocation with Pickup and Delivery	87
<i>Bradford Heap and Maurice Pagnucco</i>	
AMASON: Abstract Meta-model for Agent-Based SimulatiON	101
<i>Franziska Klügl and Paul Davidsson</i>	

Robust PI - The Simulation Robust Library Component for Pioneer 3DX	115
<i>Konrad Kulakowski and Piotr Matyasik</i>	
Secrecy Preserving BDI Agents Based on AnswerSet Programming	124
<i>Patrick Krümpelmann and Gabriele Kern-Isberner</i>	
Bid-Price Control for the Formation of Multiagent Organisations	138
<i>Marc Premm, Tobias Widmer, and Paul Karänke</i>	
Peer-to-Peer Network for Flexible Service Sharing and Discovery	152
<i>Andrii Zhygmanovskiy and Norihiko Yoshida</i>	
Agent-Negotiation of Lot-Sizing Contracts by Simulated Annealing with Part-Way Resets	166
<i>Mario Ziebuhr, Tobias Buer, and Herbert Kopfer</i>	

Workshop Track

AbstractSwarm – A Generic Graphical Modeling Language for Multi-Agent Systems	180
<i>Daan Apeldoorn</i>	
A Principled Approach for Smart Microgrids Simulation Using MAS ...	193
<i>Gillian Basso, Vincent Hilaire, Fabrice Lauri, Damien Paire, and Arnaud Gaillard</i>	
Estimating Shapley Values for Fair Profit Distribution in Power Planning Smart Grid Coalitions	208
<i>Jörg Bremer and Michael Sonnenschein</i>	
A Trust-Based Approach for Detecting Compromised Nodes in SCADA Systems	222
<i>Francesco Buccafurri, Antonello Comi, Gianluca Lax, and Domenico Rosaci</i>	
Unified Energy Agents as a Base for the Systematic Development of Future Energy Grids	236
<i>Christian Derksen, Tobias Linnenberg, Rainer Unland, and Alexander Fay</i>	
Efficient Mechanism for Aggregate Demand Prediction in the Smart Grid	250
<i>Péter Egri and József Váncza</i>	
Translating Statecharts-Based into BDI Agents: The DSC/PROFETA Case	264
<i>Giancarlo Fortino, Wilma Russo, and Corrado Santoro</i>	

Exploiting MAS-Based Simulation to Improve the Indian Railways' Efficiency	278
<i>Supriyo Ghosh, Animesh Dutta, Viviana Mascardi, and Daniela Briola</i>	
Towards a Platform for Testing and Developing Privacy-Preserving Data Mining Applications for Smart Grids	292
<i>Andrew Koster, Gabriel de Oliveira Ramos, Ana L.C. Bazzan, and Fernando Koch</i>	
Event-Driven Programming for Situated MAS with ReSpecT Tuple Centres	306
<i>Stefano Mariani and Andrea Omicini</i>	
HySoN: A Distributed Agent-Based Protocol for Group Formation in Online Social Networks	320
<i>Fabrizio Messina, Giuseppe Pappalardo, Domenico Rosaci, Corrado Santoro, and Giuseppe M.L. Sarné</i>	
Intelligent Jason Agents in Virtual Soccer Simulations	334
<i>Dejan Mitrović, Mirjana Ivanović, and Hans-Dieter Burkhard</i>	
Multi-agent Systems Applied in the Modelling and Simulation of the Protein Folding Problem Using Distributed Constraints	346
<i>Ionel Muscalagiu, Horia Emil Popa, Manuela Panoiu, and Viorel Negru</i>	
Modeling AIDS Spread in Social Networks: An In-Silico Study Using Exploratory Agent-Based Modeling	361
<i>Muaz A. Niazi, Amnah Siddiqua, and Giancarlo Fortino</i>	
Analysis of Configurations for Photovoltaic Solar Energy Production Using Agent-Based Simulation	372
<i>Jan Treur</i>	
Dynamic Allocation of a Domestic Heating Task to Gas-Based and Heatpump-Based Heating Agents	386
<i>Jan Treur</i>	
Doctoral Mentoring Track	
Improving Human-Aware Planning	400
<i>Sebastian Ahrndt</i>	
Best First Search Planning of Service Composition Using Incrementally Refined Context-Dependent Heuristics	404
<i>Johannes Fähndrich</i>	

Sequential Single-Cluster Auctions	408
<i>Bradford Heap</i>	
Decentralised Collision Avoidance in a Semi-collaborative Multi-agent System	412
<i>Sascha Hornauer</i>	
Opponent Modeling in RoboCup Soccer Simulation 3D	416
<i>Asma Sanam Larik</i>	
Policy-Based Approach for Non-Functional Requirements in Self-Adaptive and Self-Organizing Systems	420
<i>Thomas Preisler</i>	
Author Index	425

Invited Keynote: The Headaches of a Negotiation Support Agent

Catholijn Jonker

Delft University of Technology, The Netherlands

Abstract. Negotiation is a complex process as it poses challenges to the negotiator on both the emotional plane as well as on the computational plane. Human negotiators are known to leave money on the table, have trouble getting a clear view of their own preferences and those of their negotiation partner, and sometimes find it difficult to deal with their own emotions and those of their negotiation partner. In this talk I will briefly outline the Pocket Negotiator project and its prototype. I will show some solutions developed during the project and will discuss some of the open challenges. In terms of research fields, I combine means of Artificial Intelligence, Affective Computing, and Human Computer Interaction. To find out more about the Pocket Negotiator project, please visit http://mmi.tudelft.nl/negotiation/index.php/Pocket_Negotiator. To try out the prototype, please use Chrome or FireFox to visit <http://ii.tudelft.nl:8080/PocketNegotiator/index.jsp>.

About the Speaker

Catholijn Jonker (1967) is full professor of Man-Machine Interaction at the Faculty of Electrical Engineering, Mathematics and Computer Science of the Delft University of Technology. She studied computer science, and did her PhD studies at Utrecht University. After a post-doc position in Bern, Switzerland, she became assistant (later associate) professor at the Department of Artificial Intelligence of the Vrije Universiteit Amsterdam. From September 2004 until September 2006 she was a full professor of Artificial Intelligence / Cognitive Science at the Nijmegen Institute of Cognition and Information of the Radboud University Nijmegen. She chaired De Jonge Akademie (Young Academy) of the KNAW (The Royal Netherlands Society of Arts and Sciences) in 2005 and 2006, and she was a member of the same organization from 2005 to 2010. She is a board member of the National Network Female Professors (LNVH) in The Netherlands. Her publications address cognitive processes and concepts such as trust, negotiation, teamwork and the dynamics of individual agents and organizations. In Delft she works with an interdisciplinary team to create synergy between humans and technology by understanding, shaping and using fundamentals of intelligence and interaction. End 2007 her NWO-STW VICI project Pocket Negotiator has been awarded. In this project she develops intelligent decision support systems for negotiation. For more information, please visit <http://ii.tudelft.nl/~catholijn>.

Invited Keynote: Autonomous Systems Inspired by Biology

Gerhard Weiß

Maastricht University, The Netherlands

Abstract. We can currently see the rapid formation of an exciting multidisciplinary field focusing on the application of biological principles and mechanisms to develop autonomous systems (software agents and robots) that act highly flexible and robust in the face of environmental contingency and uncertainty. In this talk I will give an overview of various aspects of this field. The state of the art will be illustrated with diverse examples of bio-inspired approaches to system adaptivity, functional and structural optimization, collective and swarm behavior, locomotion, sensor-motor control, and (co-)evolution. A focus will be on representative work on biologically inspired autonomous systems done at the Swarmlab of Maastricht University, including recent research motivated by the behavior of social insects such as bees and ants.

About the Speaker

Gerhard Weiß is full professor of artificial intelligence and computer science and head of the Department of Knowledge Engineering (DKE), Faculty of Humanities and Sciences, Maastricht University. Before joining Maastricht University in 2009, he was the Scientific Director of Software Competence Center Hagenberg GmbH, Austria, and Assistant Professor at the Department of Computer Science of Technical University Munich, Germany. He received his PhD (Dr. rer. nat.) in computer science from Technical University Munich and his Habilitation degree from Johannes-Kepler University Linz, Austria. His main interests are in the foundations and in practical applications of artificial intelligence, multi-agent technology, and autonomous and cooperative systems. He is editorial board member of several journals related to his research fields, and he has been in the program and steering committees of various international conferences and workshops. He was a Board member of the International Foundation for Autonomous Agents and Multi-agent Systems (IFAAMAS) and of two European networks of excellence (Agentlink and Exystence). Professor Weiss has served as a reviewer for several national, European and international research funding organizations and has been engaged as a scientific consultant and advisor for industry. For more information, please visit <http://www.weiss-gerhard.info>.

Invited Keynote: AI-Research and the Future of the Automobil

Raúl Rojas

Free University of Berlin, Germany

About the Speaker

Raúl Rojas has been a full professor of Artificial Intelligence and Robotics since 1997 at Freie Universität Berlin. He received his PhD and *venia legendi* (habilitation) at this university. He studied mathematics and physics, as well as economics, in Mexico City. After the habilitation, he was appointed visiting professor in Viena and later professor of Artificial Intelligence at Martin-Luther-University Halle (1994-1997). His initial research was dedicated to the design and the construction of Prolog computers for Artificial Intelligence at GMD-FIRST. Today, he is working on a broad field of pattern recognition topics with special focus on neural networks and developing robots for various applications. With the FU-Fighters he won the world championship in RoboCup in 2004 and 2005. From 2006 on, he and his team have been developing autonomous vehicles, which were certified for city traffic in 2011. For his research on computer vision, Raul Rojas received the Technology Transfer Award from the Technologiestiftung Berlin (Foundation for Innovation and Technology). He was appointed a member of the Mexican Academy of Sciences in 2011. For more information, please visit http://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/pmwiki/pmwiki.php.

Similarity-Based Resource Retrieval in Multi-Agent Systems by Using Locality-Sensitive Hash Functions

Malte Aschermann and Jörg P. Müller

Clausthal University of Technology,
Institute of Informatics,
Julius-Albert-Str. 4, D-38678, Clausthal-Zellerfeld, Germany
{malte.aschermann,joerg.mueller}@tu-clausthal.de

Abstract. In this paper we address the problem of retrieving similar resources which are distributed over a multi-agent system (MAS). In distributed environments identification of resources is realized by using cryptographic hash functions like SHA-1. The issue with these functions in connection with similarity search is that they distribute their hash values uniformly over the codomain. Therefore such *IDs* cannot be used to estimate the similarity of resources, unless one enumerates the whole search space and retrieves every resource for comparison. In this paper we present a three-layer architecture and a data model to efficiently locate similar resources in linear time complexity by using *locality-sensitive hash functions*. We design the data model as an extension to distributed environments (MAS), which only need to provide at least basic resource management capabilities, such as storing and retrieving resources by their *ID*. We use a benchmark data set to compare our approach with state-of-the-art centralized heuristic approaches and show that, while these approaches provide better search accuracy, our approach can deal with decentralized data and thus, allows us to flexibly adapt to dynamic changes in the underlying MAS by distributing and updating sets of information about similarities over different agents.

Keywords: locality-sensitive hash functions, multi-agent systems, similarity search.

1 Introduction

In many real-world application contexts, such as product development [11] or medical diagnosis [13], information relevant for decision-making is distributed across systems and organizations. For instance, companies rely on efficient product data management systems (PDM) to share specifications and models with their suppliers. To date, the client-server approach is prevalent for organizing this type of systems. We found that, while it is efficient for a limited set of participants and especially if security concerns (e.g. intellectual properties) are an issue, it will not scale well for larger amounts of data distributed over larger

number of partners, which need access to minor, overlapping subsets of the data and additionally need to exchange information with participants to satisfy e.g., construction-related constraints. This problem becomes even worse because suppliers need to store a rapidly growing number of product revisions of similar, but mutually compatible parts, assemblies or even whole modules. In [11], we proposed a Product Collaboration Platform (PCP) – based upon the FreePastry [1] overlay system – supporting decentralized product development; we showed that this approach can outperform the single-server model. A very important aspect of decentralized architectures lies in the distribution of data objects by the overlay network, thus a participant can easily retrieve all the information about a specific product specification if he knows its identifier. One aspect not considered in [11] was the fact that in product collaboration environments the exact characteristics of needed parts, assemblies or even modules won't be necessarily known during development; the ability to locate specifications that are similar to a given specification can reduce development time. However, distributed semantic similarity-based search exceeds the scope of existing peer-to-peer (P2P) overlay systems such as Pastry, as it requires more computational intelligence and flexibility at the partners sides. Therefore, we are moving from peer-to-peer systems towards multi-agent systems (MAS), incorporating richer local computation models as well as more flexible modes of interaction. A practical scenario would be the collaborative development of a twin engine, which can be seen in Fig. 1. The blue highlighted activities display processes for locating specifications of similar and compatible parts or assemblies, the red activity references

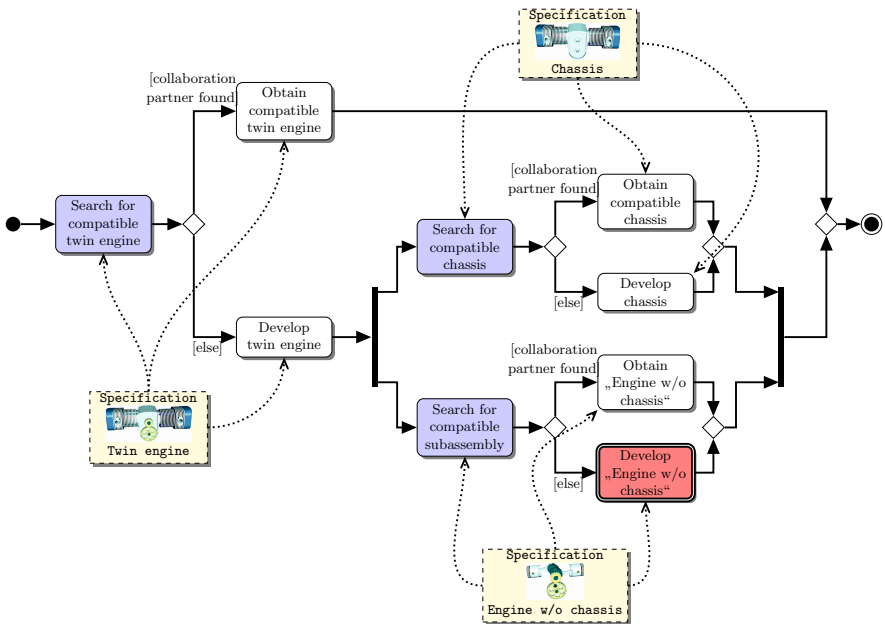


Fig. 1. Collaborative development of a twin engine (parts/assemblies from [10])

a subprocess, which, for matters of simplicity, is not shown. As shown in the activity diagram, it is – during the planning phase – reasonable to look for parts or assemblies (e.g. the chassis), which have already been developed beforehand and are similar enough to be compatible with an initial specification. This allows us to restrict the search space to similar specifications, which are, however, distributed across participants in the collaboration network. By incorporating P2P lookup techniques like distributed hash tables (DHT) into multi-agent systems (MAS), we can accomplish this efficiently in logarithmic time (e.g. binary search in trees, finger tables). However, this only works well if the hash value of the resource is known or can be trivially computed by using a previously known copy of the resource. Thus, in particular, existing methods cannot cope well with the requirements of similarity search. In this paper we design a novel general-purpose *local sensitive data model* and present an efficient distributed algorithm for locating resources similar to given *prototypes*. We compare the lookup performance of our approach with state-of-the-art centralized heuristic classification approaches, such as k-means or neural gas. In the following we will assume that an underlying decentralized infrastructure (MAS) already exists, and therefore can trivially locate resources if their exact identifier (i.e. hash-value) is known to the searching party. Thus we will elaborate how product collaboration between different participants, i.e. agents, can be optimized by introducing specialized types of agents, which dynamically keep track of similar data (i.e. resource specifications), therefore minimize the search-space and allowing to locate similar data in linear time.

This paper is structured as follows: In Section 2, we discuss related work. Section 3 provides a short introduction to similarity search by using locality sensitive hash functions. In Section 4 we present the locality sensitive data model; a qualitative and experimental evaluation is given in Section 5. Section 6 concludes with a discussion of results and outlook to future work.

2 Related Work

There is a rich body of work on grouping resources and optimizing data management with respect to availability and minimal lookup-time for certain data clusterings (e.g. by topics or similarity in general). In this section, we discuss three approaches for large-scale management of resources in distributed environments: OceanStore, Kademia and Squid.

OceanStore. The OceanStore architecture [6] is a resource management system for securely storing sensitive information in untrustworthy environments and to guarantee high availability even in very large and highly distributed networks. This is accomplished by using a technique called *promiscuous caching*, to keep as many data copies as possible distributed in the whole network at a time. Therefore OceanStore is very well suited to be used in networks where participants are scattered over large distances and resources belonging to certain users need to be made available very fast depending on the user’s current location. Though this concept comes with a high reliability and availability of data,

the identification of resources is done by using cryptographic hash functions like SHA-1 [6, p. 3]. Given the fact that it is infeasible to conclude the content of data just by knowing a hash value, one would need to completely enumerate the search space to find similar resources. Therefore this approach is not suitable for a similarity-sensitive localization.

Kademlia. The Kademlia protocol as a method for content sensitive resource management has been introduced by Maymounkov and Mazières in [7]. The authors are using lookup information which are provided by distributed hash tables to find resources efficiently. The content sensitive localization is achieved by using an xor-metric to compare distances between the *IDs* of participants and hash values of resources. Those *IDs*, whose hash values have a minimal distance to a participant, are assigned to it accordingly. Therefore requests for data objects with similar *IDs* can be answered quite efficiently. The problem with this approach also lies in the generation of *IDs*, which are – equally to OceanStore – SHA-1-based. For this reason Kademlia is not suited for grouping resources by mutual similarity of their content and does not provide an efficient way to look for similar data.

Squid. A third quite interesting approach for similarity-based data management is an extension of the P2P overlay network *Chord*. In their work Schmidt and Parashar [8] address the issue that due to the homogenous distribution of hash-*IDs*, the locality of similar resources can not be accounted for. The authors propose an extension to Chord to sustain the locality of resources [8, p. 228]. They use multi-dimensional key-spaces and thus allow lexicographic searches to neighboring resources by using *Hilbert space-filling curves*. Squid is a promising approach to locate similar data-objects in a distributed environment. However, Shu et al. point out, that “*Squid [...] partition[s] the space statically*” and that “*In Squid [...], the partitioning level needs to be decided beforehand.*” [9, p. 2]. In the context of this paper we will present a method in which no static partitioning occurs, but rather a dynamic assignment of similar resources to so called *buckets* (see Section 4) is made. We will show that this approach is far better suited for the dynamic nature of distributed environments with autonomous participants.

3 Similarity-Based Localisation

Identification, management and localisation of resources in distributed environments is typically done by using cryptographic hash functions (e.g. SHA-1) which leads to homogenous distributed *IDs* in the range of possible hash values. Although intended to minimize the probability of hash collisions, it makes finding similar resources by using such *IDs* as lookup information problematic. To find similar resources one could try to compute $\text{diff}(h^{-1}(\text{ID}_{x_i}), x_{\text{ref}}) \rightarrow \min!$ for each available resource *ID* and a given reference x_{ref} ; doing so, however, is infeasible due to the properties of cryptographic hash functions. Alternatively, one would need to request every available resource and locally compute the differences between them and the reference object. If the difference is below a given threshold

– respectively the similarity above – a candidate is found. This approach, though significant faster than computing h^{-1} , would still need quadratic time complexity ($O(n(n-1))$) to find the similarities between each of n existing resources.

In this paper we propose the use of *locality-sensitive hash functions*¹ (LSH), as originally presented by Koga et al. [5] to address this problems. In contrast to cryptographic hash functions, LSH provides collision maximising properties. Their idea is, that similar values of vectors $x \in \mathbb{R}^n$ map to the same hash-value $g(u_x) \in \{0, 1\}^k$. To accomplish this, hash functions $g_i \in \mathcal{H}$ are applied to elements of a data-set $\mathbb{X} \subset \mathbb{R}^n$, which map vectors x to hash values. Multiple functions ($l = |\mathcal{H}|$) are used in order to increase the likelihood of collisions and identify similar data-points. For using vectors of \mathbb{R}^n to represent product specifications, resources have to be transformed into vectors, by using specific characteristics (e.g. measurements) as vector attributes.

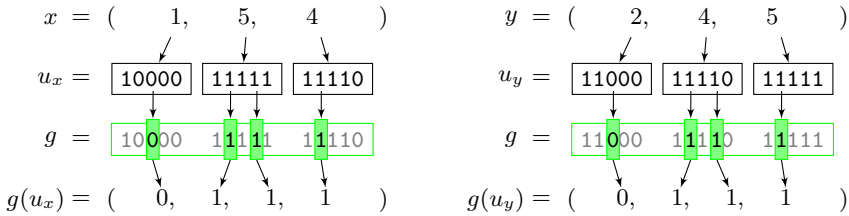


Fig. 2. Exemplary locality sensitive hashing of x and y

In a first step the vector x has to be transformed into an *unary representation*. For instance the vector $x = (1, 5, 4)$ would become $u_x = (u(x_1) \circ u(x_2) \circ u(x_3)) = (10000 \ 11111 \ 11110)$ in unary notation by concatenating x_i ones followed by $C = \max(x) - x_i$ zeros. This step is necessary to apply the hash functions g_i , which map unary vectors to binary vectors of fixed length k , denoted as $g_i : u_x \rightarrow \{0, 1\}^k$. Here, for each g_i , k indices are chosen at random of the size-fixed unary representation of x and are concatenated as follows: $g(u) = (u_{P(1)} \circ u_{P(2)} \circ \dots \circ u_{P(k)})$, with set P , containing a random permutation of indices $\{1, \dots, k\}$.

Each vector is organised in sets called *buckets*. A bucket, for example B_h , is uniquely identified by a hash value $h \in \{0, 1\}^k$ and contains all pairwise collisions (i.e. potential similarities) of vectors which share the same LSH-value h .

Taken the prior chosen value of $x = (1, 5, 4)$ and additionally $y = (2, 4, 5)$ with a single hash function g and an exemplary index permutation of $P = (3, 7, 9, 12)$ (implying $k = 4$) the resulting hash values would be $g(u_x) = g(u_y) = (0, 1, 1, 1)$ (see Fig. 2). Due to this hash collision, x and y belong to the same bucket $B_{(0111)} = \{x, y\}$, identified by the hash value of their collision ($ID(B_{(0111)}) = (0, 1, 1, 1)$). This would indicate that x and y are similar to each other.

LSH-heuristics are primarily used for an approximate – but quite efficient – computation of k -nearest-neighbor graphs in linear time [5, p. 116], where the

¹ We use the term of locality in the sense of semantic similarity, rather than geographic location.

time complexity amounts to $O(nl|B|) < O(n^2)$. Beyond that, this technique can be used to locate similar data points to a given *prototype*, which can be seen as a search query to look for. The goal is to find data points (i.e. *candidates*), **near** a given prototype p , of a finite data set $\mathbb{X}^{\text{cand}} \subseteq \mathbb{X}$. Candidates s are *similar* to a search request p iff

$$s \in \mathbb{X}^{\text{cand}} \Leftrightarrow \exists g, g' \in \mathcal{H} : g(s) = g'(p) \vee g'(s) = g(p).$$

This means, that all s – for which at least one hash function of \mathcal{H} leads to a collision with p – are candidates and therefore similar to p . From the set \mathbb{X}^{cand} , which is significant smaller than \mathbb{X} , data points worth further consideration can be selected using exact measuring methods due to the quite efficient elimination of irrelevant data in the previous step.

4 Locality Sensitive Data Model (LSDM)

We propose a three layer architecture based on LSH to address the problems as previously explained in the introduction, i.e., how to efficiently locate similar resources in distributed multi-agent environments, where the identification commonly takes place via cryptographic hash functions. We propose an extension to existing MAS-based resource management systems (e.g. [12], [4]) with the intention of providing a highly flexible method to add similarity-based localization to other platforms and to reduce unnecessary complexity.

Architectural Design. The actual extension consists of agents, managing lists of mutually similar resources. These agents, which we further call *locality-sensitive agents (LS-agents)*, are not only responsible for sharing their own resources, but also for managing similarity lists and keeping these up-to-date. As shown in Fig. 3, the LSDM extends a classic two-layer architecture, consisting of a resource and agent layer, with a third layer containing the LS-agents. The technique, presented in the previous Section 3 (finding similar vectors and organizing them in buckets by using LSH functions) will be modified in such a way that it can be built upon MAS.

The goal is to link reference specifications (search queries) q to sets of similar resources. To minimize the workload and to increase the availability of these sets, it is not advisable to entrust them to one single agent. Rather we distribute the buckets among available agents. This can be achieved by viewing buckets as resources and using the same cryptographic hash function f_X which is used to identify agents responsible for certain resources and to identify the resources themselves. If this hash function f_X (e.g. SHA-1) is applied to bucket-IDs, they can be treated like ordinary resources and the responsible agent can easily be determined by using the underlying distributed resource management system. For this assignment in detail, see Fig. 4. Here the LS-agents hold – beside their own ID – sets of similar resources S_h . Taking into account that storing whole resources in S_h is not desirable for reasons of data protection and possibly large resources, we only store their cryptographic hash values (which sufficiently identify resources) in S_h .

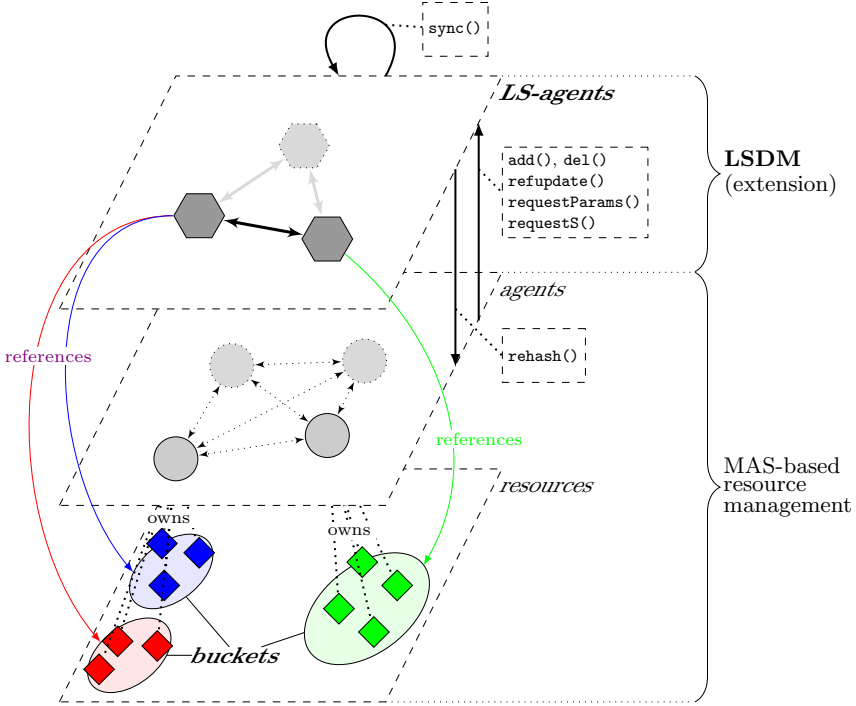


Fig. 3. Three-layer architecture of LSDM

Identification of Similar Resources. Assuming a reference specification $s \in \mathbb{R}^n$ exists, we can locate similar resources using the above described approach. At first we need to compute all LS-hash values h to determine bucket IDs, in which collisions (i.e. similar resources) can be found and put them into one set B_s : $B_s = \{h \mid h = g(s), \forall g \in \mathcal{H}\}$. Then we can identify the responsible agents by determining $ID_{\text{agent}}(h) = f_X(h) \forall h \in B_s$ and request their similarity sets S_{h_i} by using the underlying resource management system (e.g. issuing a $\text{get}(ID_{\text{agent}}(h))$ command). If we merge these sets S_{h_i} into one combined set $S = S_{h_1} \cup \dots \cup S_{h_n}$ we will get every ID of resources similar to the reference specification s .

Requirements to the LSDM. Regarding the design of our data model, the following requirements are of great importance: 1. If information about similar objects is passed to the LSDM, it has to accept and process them in any case. 2. Similar resources (if they exist) to any given reference specification (i.e. prototype) have to be returned to the user. 3. Regularly modification of resources needs to be taken into account. 4. Resources, which are removed from the underlying data model, also needs to be removed from the LSDM. 5. In decentralized environments sign-outs or even unforeseen malfunctions of agents can not be ruled out. Therefore appropriate countermeasures need to be in place to prevent loss of information. These issues can be addressed as follows:

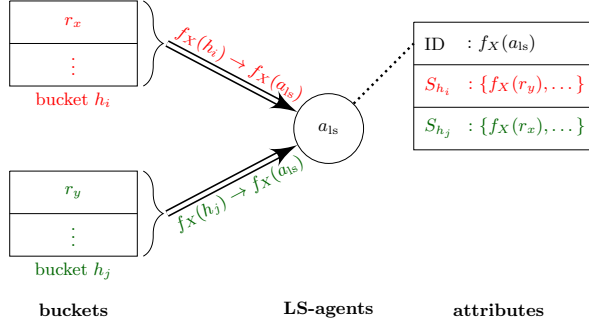


Fig. 4. Assignment of buckets to LS-agents by using a cryptographic hash function f_X

1. **Publication of Resources.** Initially every agent a , which publishes a resource r to the MAS, needs to have the necessary LSH parameters (i.e. k, l, \mathcal{H}), for which it issues the `requestParams()` command to one of the existing LS-agents. (Since the LSH parameters have to be the same throughout the LS-agents, it can ask any of them.) Then it can compute $h_i = g_i(r), \forall g_i \in \mathcal{H}$ ($i = \{1, \dots, |\mathcal{H}|\}$) to get a set of LSH-values h_i , which identify similarity lists, managed by LS-agents. These lists can now be requested from the MAS by viewing h_i as regular resources and thus get the responsible agents by $ID_{LSagent_i} = f_X(h_i)$. In the last step the publishing agent calls `add(ID_{LSagent_i}, h_i, f_X(r))`, which tells $ID_{LSagent_i}$ to add the hash value of r to its similarity list S_{h_i} . Each LS-agent also stores, additionally to ID_r , the ID_a of its publisher, which is useful to push reshash requests to agents. (See Fig. 5 and 5. *Exception Handling* below.)
2. **Localisation of Similar Resources.** To find similar resources, according to a reference specification s , the agent requests the LSH parameters from any LS-agent via `requestParams()`. If it already received those, this step

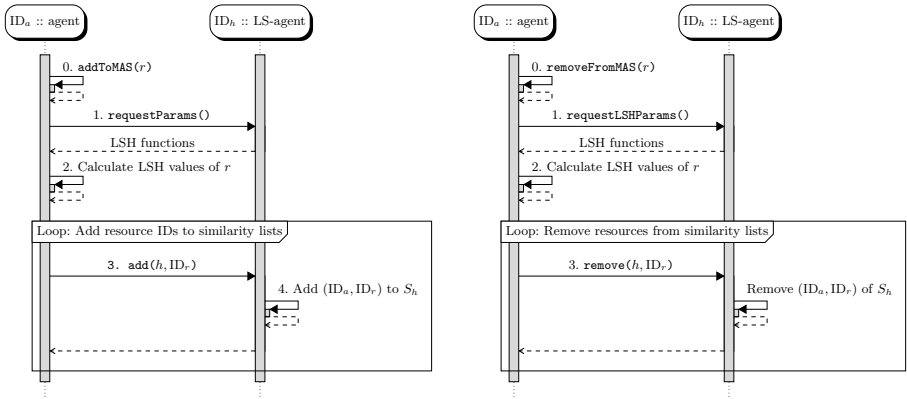


Fig. 5. The process of adding resources to LSDM (left) and removing them (right)

can be omitted. The agent needs to compute the set of LSH values B_s to get all LS-agents $ID_{LSagent_i} = f_X(h_i)$, with $h_i \in B_s$, responsible for bucket h_i . Thereupon the similarity lists S_{h_i} can easily be received by calling `requestS($ID_{LSagent_i}, h_i$)` to the corresponding LS-agents. The actual resources, which are similar to s , are retrievable by their $ID \in S_{h_i}$ by using the methods of the underlying MAS.

3. **Modification of Resources.** If resources are modified ($r_{old} \rightarrow r_{new}$), it will have an impact on their similarities to other existing resources. Therefore the assignments of outdated ($f_X(r_{old})$) and new ($f_X(r_{new})$) resource ID s to similarity lists S need to be reviewed and – if necessary – updated. To accomplish this, the modifying agent calculates $B_{r_{old}}$ and $B_{r_{new}}$ (analogue to above) and tells every LS-agent derived from $B_{r_{old}}$ to remove the similarity references (`del()`) and every LS-agent derived from $B_{r_{new}}$ to add those (`add()`).
4. **Removal of Resources.** Resources r , which are removed from the MAS, need to be removed from the LSDM as well. Again, an agent needs to have the LSH parameters, which it can retrieve via `requestParams()`. It then computes all buckets which contain hash values $h_i = g_i(r), \forall g_i \in \mathcal{H}$, identifying similarity lists S_{h_i} . After that, the agent tells the responsible LS-agents to remove all references to ID_r from their similarity lists (`del()`). (See Fig. 5)

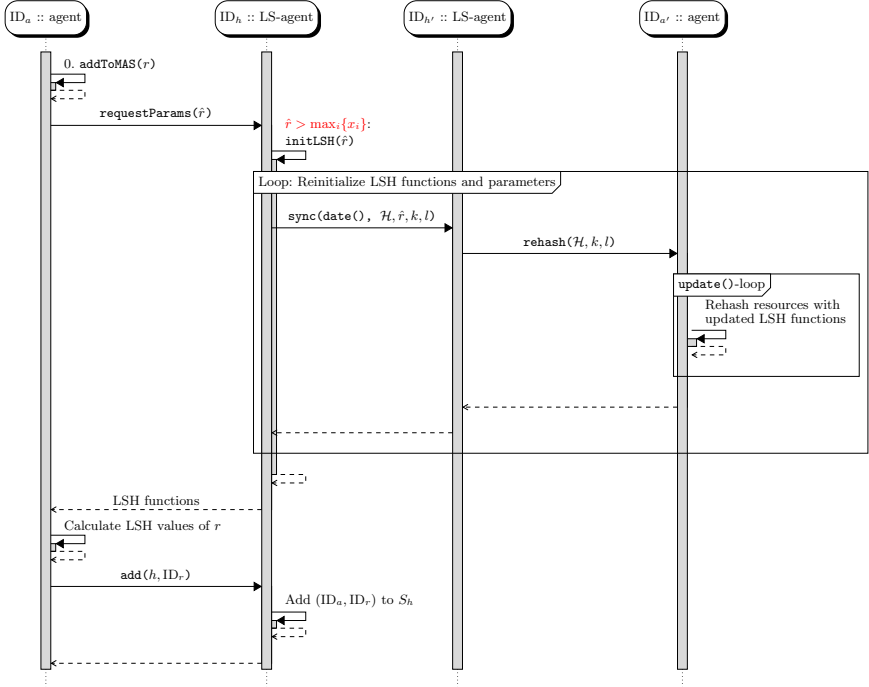


Fig. 6. The process of adding resources if the maximal vector attribute is exceeded

5. **Exception Handling.** One aspect which arises from using LSH functions is, that once hash functions of \mathcal{H} are defined, they constrain the size of possible vector attributes by $\max_{x \in \mathbb{X}}(x_i)$. If a new vector $y \in \mathbb{R}^n \setminus \mathbb{X}$ needs to be hashed, we have to *rehash* every existing resource in \mathbb{X} and rebuild and redistribute the buckets. In this case every LS-agent tells each agent a , which can be found in its similarity lists, to `rehash()` and republish their resources. (See Fig. 6)

Additional issues are malfunctioning or suddenly disconnecting LS-agents. Therefore, to prevent loss of similarity lists, we need to have backups among the agents of the MAS. We propose to have additional (backup) LS-agents to counteract these issues. Similar to [8] we can apply xor-metrics to LS-agent's IDs to determine *neighboring* agents and let them take over, if actual LS-agents are not available anymore.

5 Evaluation

The LSH technique underlying our data model is the central aspect of this paper, and its overall effectiveness depends on the accuracy of LSH's approximation. Therefore, we focus our evaluation on a qualitative analysis of our data model and on experimental comparisons between our approach and state-of-the-art clustering techniques on standard benchmark data. Here we want to show, that our approach can compete with well known clustering techniques and – moreover – prove its usefulness in decentralised environments, such as MAS, by providing a fair approximative classification of the available and shared resources in order to minimize the relevant similarity-search-space beforehand.

Qualitative Analysis. For the distributed management of similarity information in multi-agent systems, our proposed data model LSDM – as part of a three layer architecture – can be an effective approach compared to centralized clustering of resources, by distributing the workload of clustering and communication over various agents. By introducing the concept of *locality sensitive hash functions*, we distributed the main workload of computing similarity information over the participating agents (i.e. collaborating partners, willing to share resources) and assigned the management of *similarity lists* to multiple LS-agents. Here the time complexity for each agent to LS-hash its resources amounts to $O(nl|B|)$ (see [5, p. 116]), where n denotes the number of resources, l the number of LS-hash functions and $|B|$ the bucket size. Therefore this process is of linear computational complexity (see Section 3). As this step only needs to be done initially or if a rehashing is needed in exceptional cases (see Section 4), this approach has very little impact on the overall computation time of each agent. The management overhead of buckets, induced by the LS-agents is minimal, as these agents only act as *'yellow pages'* for search queries. This method also incorporates the idea of multi-agent systems, by providing a more fault tolerant and load balanced way of managing buckets compared to a centralized clustering instance and avoiding performance penalties induced by *bottlenecks* or *single-point-of-failures*. Without the need for such an instance, our

data model can be used in contexts with high volatility of available agents if precautions – in terms of backup LS-agents – are taken.

Experimental Setup and Results. In our experimental scenario we measure the accuracy of finding similar elements from a database, containing disjoint classes of similar elements. As a benchmark data set for our evaluation, we chose the Wisconsin Diagnostic Breast Cancer database (WDBC), a well known benchmark set of clinical cases [13], which we transformed into integer-based vectors, in order to use it for our data model. The data set consists of 699 data points (but due to 16 incomplete results we could only use 683), each described as an 11-dimensional vector containing two identifying attributes (sample ID and classification). Therefore we used nine attributes as a data vector and the remaining two to assess the accuracy of our similarity approximation by using the Rand index.

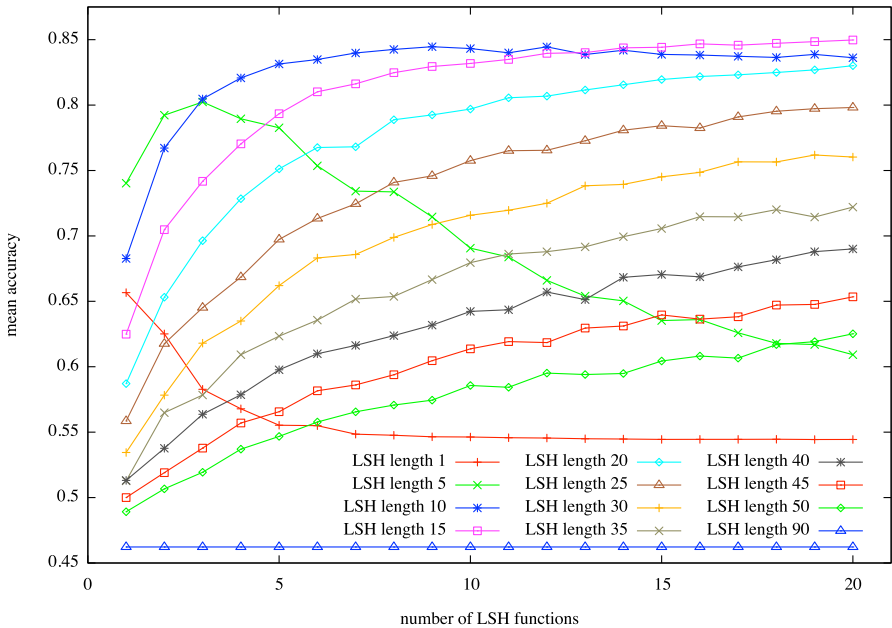


Fig. 7. Accuracy of LSH against the number of used LS-functions

In our experiment we only focused on the accuracy of LSH and measured the impact of different hash output lengths and number of used hash functions to the resulting accuracy of classification. We chose output lengths of 1 to 5, from 5 to 90 bits in steps of five and 1 to 20 hash functions. For each combination of number of hash functions and output length, we ran the LSH classification 100 times to obtain a good mean accuracy. In each iteration we randomised the hash functions to minimise errors due to bad functions. As shown we plotted the average accuracy against the number of used functions Fig. 7 and output length Fig. 8. (The omitted samples lied in between those shown and were removed for a better overview.)

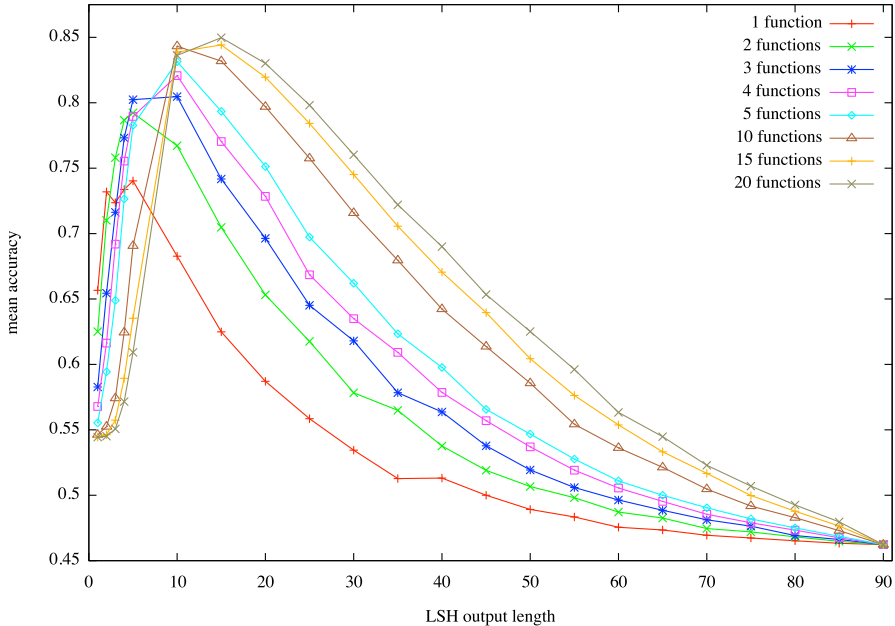


Fig. 8. Accuracy of LSH against the output length of used LS-functions

What our results show is that the accuracy of LSH’s resource classification depends on two important parameters: The number of used hash functions and the unary output length of them. Regarding the output length our experiment shows, that by increasing the output length towards the allowed maximum ($vmax_{wbc} = 90 = \text{len}(x) \max_{x \in \mathbb{X}}(x_i)$), the accuracy decreases continuously until it reaches a minimum of 0.46 (Fig. 8) for our chosen data set, which can also be seen in Fig. 7 for the output length of 90. By combining the results for both parameters, we measured an accuracy of 85.0% with a standard deviation of 0.7 for an – in our case – optimal parameter choice of 20 hash functions with a length of 15. Regarding Fig. 7, one additional fact worth mentioning are the two curves for LSH lengths 1 and 5. Here we assume, that by increasing the number of hash functions, the amount of false-positives grows significantly faster compared to the other curves, due to the very short output lengths. Therefore the results for 1 and 5 can be treated as anomalies.

Table 1. Accuracy of LSH compared to state-of-the-art techniques for WDBC classifications

	LSH	k-Means [3]	Supervised Batch NG [3]	C4.5 decision tree [2]
Accuracy in %	85.0	93.6	94.7	96.0
StdDev	0.7	0.8	0.8	–

In Table 1 we confronted our results for the WBCD with other, comparable, experiments (see [3], [2]). As can be seen, heuristics like *k-Means*, *Neural Gas* and *C4.5* outperform our approach in terms of accuracy by roughly 10 – 20%. Here we would like to point out, that these heuristics operate centralized with complete knowledge of the available data set. Our data model in contrast, is able to compute bucket *IDs* without concrete knowledge of any existing resource, and purely relies on the approximation done by previously chosen LS-hash functions. Furthermore, every single agent only needs to know his own resources and does not need to exchange any information with other agents, except for buckets, containing cryptographic hashed values of resources. Due to the fact that the accuracy also considers elements that are similar but not located, it is possible to obtain them by increasing the number of hash functions and thus making it more likely for them to be found. Although this approach increases the set of false-positives, it can, by gradually extending $|\mathcal{H}|$, help to get more similar resources. Therefore, given that our main goal was to minimize the search-space for similar resources, an seemingly lower accuracy can significantly reduce the costs of comparing possible candidates, if the overhead, induced by non-similar resources, would still be orders of magnitude smaller than the whole search-space.

6 Conclusion and Outlook

The central aspect of the locality sensitive data model (LSDM) is the extension of an existing multiagent system (MAS), used in decentralized collaborated product development, to efficiently locate similar resources. The main contributions of this paper are twofold. First, we proposed and evaluated the incorporation of collision maximizing hash functions into existing agent-based resource management systems. Here the goal was to avoid a centralized approach, by using one single agent to manage sets of similar resources, but instead distribute these sets over multiple agents to minimize the management overhead, bottlenecks and single-point-of-failures. By using locality sensitive hash functions it is possible to compute the buckets, containing *IDs* of similar resources, in linear time and distribute them over LS-agents which are responsible for them.

Second, we showed in our experiments that that choose suitable LSH parameters (number of hash functions and output length) is important for achieving good results. Although we discovered that our approach is unable to achieve the same accuracy results as centralized state-of-the-art clustering and classification heuristics in terms of overall accuracy, we managed to reduce the search-space for similar resources significantly. Here further research is needed to conclude if the LSH parameter can be determined beforehand, as prior benchmarking and computation of optimal parameters is often not practical.

Our current approach faces several limitations: First we pointed out, that only vectors of \mathbb{N}^n are suitable for LS-hashing. Therefore, as mentioned in Section 3, resources have to be transformed into integer vectors, by using specific characteristics of data objects (e.g. measurements) as vector attributes. Real-valued

vectors for example could be easily transformed by using a fixed length of decimal places and multiplying them with a constant scalar value, but this would drastically limit the overall precision of the data set. Therefore it is reasonable to conduct further research on how to express product specifications as vectors or use different approaches in this context. Second, we have chosen the WDBC dataset for experimental evaluation in the absence of publicized benchmark sets of product models. While our method itself is general purpose, this may restrict generalizability of the experimental results to similarity search of product models. Research is needed towards creating appropriate benchmark datasets. Finally, we need to explore ways of further improving the accuracy of LSH-based search e.g., by modifying the random generation of LS-hash functions to weight more relevant vector attributes.

References

1. Druschel, P., Engineer, E., Gil, R., Haeberlen, A., Hoyer, J., Hu, Y.C., Iyer, S., Ladd, A., Mislove, A., Nandi, A., Post, A., Reis, C., Sandler, D., Stewart, J., Singh, A., Zhang, R.M.: Freepastry (2012), <http://freepastry.org/FreePastry/>
2. Hamilton, H., Cercone, N., Shan, N., University of Regina. Dept. of Computer Science.: RIAC: a rule induction algorithm based on approximate classification. Tech. rep. (1996)
3. Hammer, B., Hasenfuss, A.: Relational neural gas. In: Hertzberg, J., Beetz, M., Englert, R. (eds.) KI 2007. LNCS (LNAI), vol. 4667, pp. 190–204. Springer, Heidelberg (2007)
4. Kelash, H.M., Faheem, H.M., Amoon, M.: A multiagent system for distributed systems management. World Academy of Science, Engineering and Technology 11, 91–96 (2007)
5. Koga, H., Ishibashi, T., Watanabe, T.: Fast hierarchical clustering algorithm using locality-sensitive hashing. In: Suzuki, E., Arikawa, S. (eds.) DS 2004. LNCS (LNAI), vol. 3245, pp. 114–128. Springer, Heidelberg (2004)
6. Kubiatowicz, J., Binde, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B.: Oceanstore: An architecture for global-scale persistent storage. In: Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2000 (2000)
7. Maymounkov, P., Mazières, D.: Kademia: A peer-to-peer information system based on the XOR metric. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 53–65. Springer, Heidelberg (2002)
8. Schmidt, C., Parashar, M.: Flexible information discovery in decentralized distributed systems. In: Proceedings of the 12th High Performance Distributed Computing (HPDC), pp. 226–235. IEEE Computer Society (2003)
9. Shu, Y., Ooi, B.C.: lee Tan, K., Zhou, A.: Supporting multi-dimensional range queries in peer-to-peer systems. In: Fifth IEEE International Conference on Peer-to-Peer Computing, pp. 173–180. IEEE (2005)
10. Siemens: PLM, JT2Go, 2 Cylinder Engine (typical or “shattered” JT file) (2012), http://plm.automation.siemens.com/en_us/products/teamcenter/lifecycle-visualization/jt2go/downloads/index.shtml

11. Stiefel, P.D., Hausknecht, C., Müller, J.P.: Using ontologies to support decentral product development processes. In: Fischer, K., Müller, J.P., Levy, R. (eds.) ATOP 2009 and ATOP 2010. LNBIP, vol. 98, pp. 114–129. Springer, Heidelberg (2012)
12. Vilà, P., Marzo, J.L., Calle, E., Fàbrega, L.: Multi-agent system co-ordination in a distributed network resource management scenario. IEEE (2005)
13. Wolberg, W.H., Street, W.N., Heisey, D.M., Mangasarian, O.L.: Computer-derived nuclear features distinguish malignant from benign breast cytology. *Human Pathology* 26, 792–796 (1995)

A Multi-agent Flooding Algorithm for Search and Rescue Operations in Unknown Terrain

Matthias Becker, Florian Blatt, and Helena Szczerbicka

Simulation and Modelling Group
Faculty of Electrical Engineering and Computer Science
Leibniz University Hannover, Germany
{xmb,blatt,hsz}@sim.uni-hannover.de

Abstract. In this paper we will introduce a new multi-agent algorithm for the use in search and rescue scenarios for exploration of unknown terrain. This method combines the concept of exploration from the flood algorithm and the path optimizing features of the ant algorithm. The first part leads to a fast exploration of the unknown terrain, while the second part constructs short paths from points of interest back to the base. Together this enables the starting of rescue operations parallel to the ongoing search. We demonstrate the feasibility of our approach by agent-based simulations. The simulations show, that our approach is comparable in speed and quality with already existing algorithms, delivering the additional benefit of short paths to points of interest, and adhering to the inherent limitations of these kind of scenarios.

1 Introduction

A modern search and rescue scenario poses various challenges for the teams doing the search and rescue operations. The primary goals of these teams are the fast exploration of the disaster area and the fast rescue of the victims. Since in many cases the searching procedure endangers the search team members, which may be the case in unstable buildings after an earthquake or in the wake of a nuclear disaster. This lead to the introduction of robots in these kinds of scenarios.

A typical search and rescue scenario consists in two parts: the search part and the rescue part, as defined by Murphy et al. in [6, p. 1152]. Usually algorithms will only solve the first part of the problem, while ignoring the second part. There are various search and rescue robots available today, that are capable of finding one or more victims or generally one or more points of interest [9,2,7]. There are also various ideas for algorithms, that solve the search part of the problem [4,5].

Ferranti et al. state in their paper [4], that they want to improve *Brick and Mortar* algorithm to maintain communication channels between the agents in the field and the rescue team in the base. So that if an agent finds a victim, the position of said victim is communicated to the headquarter as soon as possible. Another challenge, that they want to overcome is the possibility of a changing

map. If, for example, walls collapse and open new pathways or close already visited paths, recent information may not be valid anymore, hindering the search and the rescue process. The same authors propose another algorithm [5], which tries to create a communication network by using active wireless sensor nodes. While this would enable the agents to communicate among each other and directly with the base, this approach is limited by the amount of sensor nodes needed to be carried by the various agents.

Our goal is to create an algorithm, that is able to communicate directly with the base without the use of wireless sensor nodes and add the possibility to monitor the environment and act on changes in said environment.

The functionality of the *Brick and Mortar* algorithm is not usable to realize this goal, as the agents strictly avoid already visited terrain and will come to a stop, which in turn will stop the whole algorithm, if no agent moves anymore. For this reason other path finding algorithms have been evaluated. These algorithms work quite well, as long as the path has only one starting point and one end point. The conventional ant algorithm [3] and also advanced algorithms like Scout Ant Cooperation [11] can only find one end point, ignoring other prospective end points and optimizing only this single path. Richard Bellman introduced a cellular automaton in [1], which can compute the shortest path between a starting point and various end points. This approach has been adapted, allowing to search for multiple end points and, additionally, allowing to compute paths from the starting point to these explored end points. Adding the behavior of the standard ant agent to our algorithm grants the ability to optimize the found paths, as the agents will usually not find the shortest path on the first try.

In the following section we define our scenario including its limitations. The third section provides a short overview of current algorithms, while we introduce our new algorithm in the fourth section. The fifth section presents our experimental setup and Section 6 present a discussion of the results, followed by a conclusion in Section 7.

2 Definition of the Scenario

First we need to define our scenario and our preliminaries for our new algorithm. Real search and rescue robots work under different conditions in the field. This means, that there will be hardware limitations, which may affect the sensory input of the agents or can deny the use of a sensor straight away. The first sensor which is compromised most of the time is the Wifi sensor, as wireless networks become quickly saturated with no way to establish a priority over which information should be send or is received [6, p. 1168]. As a consequence the agents are not able to use direct communication between themselves or with the base. This lack of direct data transfer between the various agents forces them to communicate via indirect methods, such as markings on the floor or the exchange of RFID tags [4,10,5]. In many situations the Wifi sensor can only handle a direct communication between two agents, who are currently in a line of sight. The exchange of RFID chips is only limited by the amount that can be

carried by the agents. Usually the small size of the chips should allow to carry a large quantity of them.

The map is divided into various cells of the same size. This definition is akin to the definition of the cells from the *Brick and Mortar* algorithm [4]. Additionally the range of the RFID sensors should be kept in mind while choosing the size of the cells, as the agents should be able to read the chips from the adjacent cells. Each cell may contain either free space, a wall, the starting point or a point of interest. These points of interest are the end points for the algorithms and will depict our victims. They are equally distributed through the whole map. We assume, that these points of interest are static and do not change their position at this time.

An agent is able to leave markings in the cells that are traversable. Other agents can read these markings and can update the contained information.

3 State of the Art

As mentioned above, there are already some algorithms available to solve the search part of the problem. These algorithms include the *Brick and Mortar* algorithm [4], the *HybridExploration* algorithm [5], and also the *Multiple Depth First Search* (DFS) [4]. These three algorithms are able to search an unknown map for multiple points of interest, but are not able to create a path to at least one of these found points or return to the base by themselves.

Another recent related work includes a coverage algorithm [8]. The *StiCo* algorithm from Ranjbar-Sahraei et al. uses ant agents to monitor unknown terrain. The various agents move in circles and try to keep their area out of the circles of the other agents, thus spreading themselves out. As soon as each agent can move in an area of its own the algorithm has reached a stable configuration. This approach is not applicable to our problem, as the algorithm will leave some parts of the map unexplored as long as a stable configuration is reached. The authors formulated the maximum possible coverage of a square map with a set of disjoint identical circles with 78.5%.

The *Brick and Mortar* algorithm is our bench mark algorithm used as comparison to our approach, as it has good coverage and run time properties. The agents use the markings in the cells to tell which cells already have been visited. It differs between marking one cell as closed or as explored, but maybe still needed for movement. In this way the algorithm tries to spread the agents as wide as possible. The only drawback are obstacles in the field, which need a special loop detection. Without this detection, some agents may try to circuit the perimeter of the obstacle more times than it is necessary, resulting in a longer run time. The algorithm will stop as soon as the agents can not move anymore.

The *HybridExploration* algorithm is an enhanced version of the *Brick and Mortar* algorithm. It tries to optimize the loop detection of the base algorithm by transplanting the actual movement of the agents into a virtual plane. For this to work, the agents need to use active sensor nodes, which will be used instead of

simply marking the free cells. If the agent suspects a loop, it will start a *virtual* agent, that will circuit the obstacle via the active sensor nodes.

The reason why the older *Brick and Mortar* algorithm instead of the newer *HybridExploration* algorithm was chosen as a benchmark is simply, that the enhanced version relies on active sensor nodes. This works in rather small environments, where there is no detraction to the radio signal and where the agents are able to carry enough sensors with them to fill the whole map. Otherwise *Brick and Mortar* may be slower but it does not depend on these two preliminaries.

4 The Multi-agent Flooding Algorithm

In this section we introduce our new algorithm. It is named after the basic principles, that it embodies: the flooding aspect and the multi-agent nature. The algorithm is based on a single type of agent, which can work alone or can also work in a team to increase the speed of the search. Another advantage is, that the agents are based on the simple ant agent [3,4], therefore they will not need a great computing capacity.

The main goal of the algorithm is to search an unknown terrain for points of interest and after finding one of these points, to return with the information back to the starting point. This would allow the gathering of information at the base and even a sharing of information between the different agents, as they visit the starting point again. Additionally the path, that the different agents take to return to the base could also be optimized through the use of new information gathered by other agents. This will allow the rescue teams to act immediately on the information and in turn start the rescue phase parallel to the search phase. Other algorithms will try to finish the search phase first and will only start the rescue phase after they have gathered the whole information.

We did not create a special condition for the termination of the algorithm, as this could interrupt the monitoring of the terrain. The return to the starting point can be used to stop the single agents one by one as they revisit the base. This was a design decision, as the search for victims should continue as long as possible. Victims could move and change their position as long as they are not found or the already explored terrain could change and open up new places, that still need to be searched. Which would not be realizable, if the algorithm would ignore already explored space completely. This concept also offers the advantage of a possible battery exchange, repair works, or simply general maintenance of the robots at the base before they return to their search.

The agent has two modes: the search mode and the return mode. Each agent starts in the search mode.

4.1 Search Mode

A single agent can move from his position in eight directions in the matrix similar to the usual ant agent. The similarity to the ant agent has been chosen, so that the agents can be kept as simple as possible. Should one of these cells contain a

point of interest, then the agent prefers this cell. Otherwise it prefers unmarked cells and moves in the same direction it faces with a high probability, currently 95%, or it will change the direction with a corresponding lower probability, in this case 5%. After the movement the agent marks his new position in the matrix with its step counter, which starts at one and is incremented by one with each step, this is the only data, that the agent will need to save. If all neighboring cells are already marked, it chooses the cell with the highest marking with a 95% probability or any other cell otherwise. If its step counter is smaller than the marking in the current cell, then the agent overwrites this marking with its step counter, which will “shorten” the way for any agent returning via these cells back to the starting point.

4.2 Return Mode

If the agent finds a point of interest and moves into the respective cell, it will switch its mode to the “Return Mode”. In this mode it does not mark any cells and returns to the starting point via the markings written in the cells of the matrix during its own search mode or that of other agents. Starting from the point of interest it scans all neighboring cells for the lowest marking and moves into the cell with the minimum mark. The agent repeats this pattern until it has found its way back to the starting point, where the step counter is reset to one. This also triggers the mode switch again, putting the agent back into the “Search Mode” once more.

If the agent happens to find multiple points of interest at the same time in the adjacent cells the first point in the search pattern will be found, whereas the neighboring cells are searched from the upper left to the lower right.

5 Experimental Simulation Setup

We have chosen five different scenarios for the simulation of our algorithm. Two maps with a size of 1000×1000 cells and three maps with the size of 500×500 cells. The smaller maps contain ten points of interest which are uniformly distributed in the free space, while the bigger maps have 15 points of interest, also uniformly distributed. Each map contains one starting point and its position was randomly chosen from the number of cells, that adjoin the border of the map. One of the smaller and one of the bigger maps where simple plains without any obstacles, while the rest of the testing scenarios where modeled after house interiors or a street of houses. The Figures 1, 2, and 3 depict three of the five test cases used in this simulation. The simple plains were chosen as a standard test case. Each algorithm should be able to cover the whole map and find all points of interests in this case, without having to handle any obstacles. Figures 2 and 3 were chosen to simulate urban scenarios, showing an outline of a single house or even the outlines of multiple houses. Resulting in targets that can be reached via multiple paths or even only through a single path. The last test case, figure 1, was especially chosen to try to trap the agents in the small paths.

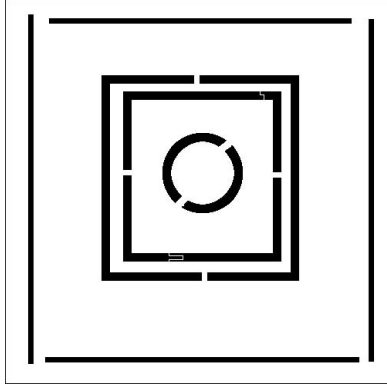


Fig. 1. Plain with obstacles, 500×500 cells

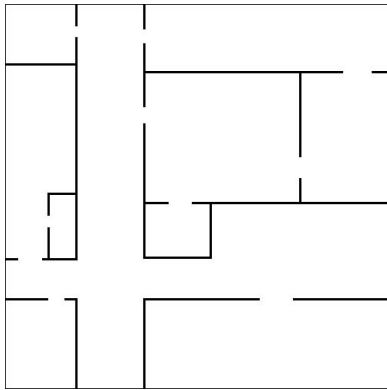


Fig. 2. House, 500×500 cells

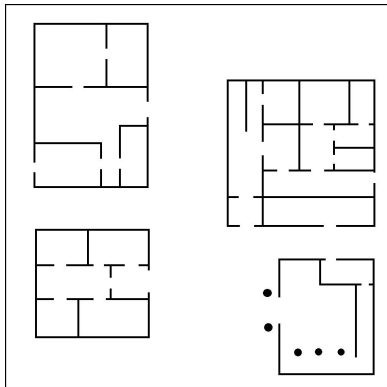


Fig. 3. Street of houses, 1000×1000 cells

As mentioned above we used the *Brick and Mortar* algorithm as a benchmark for the comparison of the results of our algorithm. Our tests started with 30 simulations with one agent and the number of agents was incremented by one after these 30 iterations until we stopped after the last simulation with 20 agents.

5.1 Simulation Experiments

The time in these experiments was measured in steps. Where one step is the time, that each agent needs to move from one cell to another, including the time that is needed to decide where to move. To compare the results of the two algorithms, as one will terminate when no agent is able to move anymore and the other one has no direct method of termination, we decided to let both run until 95% of the map were explored and 95% of the points of interest were found. The number of steps each algorithm needs to reach these boundaries is proportional to the run time of each algorithm. To keep the simulations manageable each map has a maximum run time depending on the size of the map. This run time is equal to the number of all cells in a map, resulting in a cap of 250.000 steps for the 500×500 test cases and 1.000.000 steps respective for the 1000×1000 test cases, as the size of the cap should allow one agent to explore the whole map once.

The length of the paths, that the *Multi-Agent Flooding* (MAF) algorithm has found was ignored in this case, as the *Brick and Mortar* algorithm is not able to create paths for a comparison.

6 Results and Discussion

The following diagrams (Figures 4-8) represent the results of our simulation experiments. Each diagram depicts the number of agents on the X-axis and the number of cells on the Y-axis. Each point in the line shows the average of 30 simulation runs with the corresponding number of agents. There are two lines which belong to each algorithm. One line shows the average number of steps the algorithm needed to find 95% of the points of interest and explore 95% of the map or reach the boundary of the map. The other line shows the average number of explored cells at the termination of the algorithm.

The results show, that for the smaller test cases a minimum number of five or six agents is needed to explore the whole map. Adding additional agents will only improve the search time. Interestingly two agents were able to explore most of the map in the bigger test cases and they were also able to find 95% of the points of interest, terminating the algorithm before the cap of 1.000.000 steps was reached. Whereas the flood algorithm needed about five agents in the smaller maps to find 95% of the points of interest and explore 95% of the viable cells, before the algorithm could reach the step counter boundary. The depicted graphs also suggest, that the gain from adding additional agents slowly tapers off. Raising the question, when the cost-benefit ratio equals zero or drops into the negative range. This is especially visible in the plain test cases, shown in Figures 6 and 8.

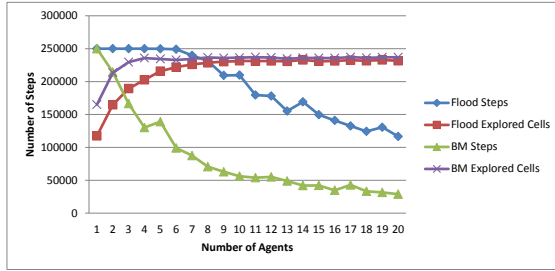


Fig. 4. Results House, 500×500 cells

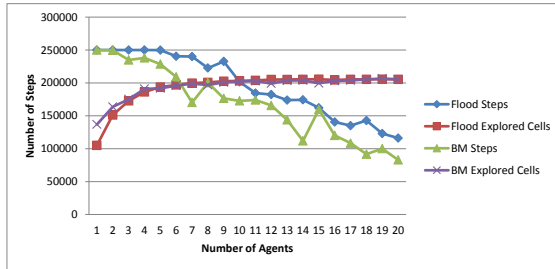


Fig. 5. Results Obstacles, 500×500 cells

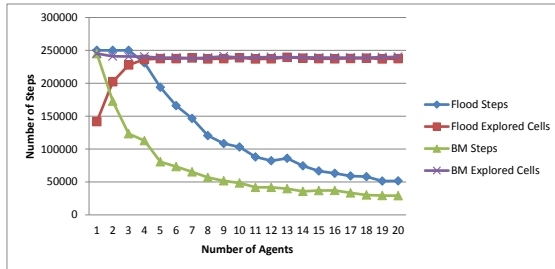


Fig. 6. Results Plain, 500×500 cells

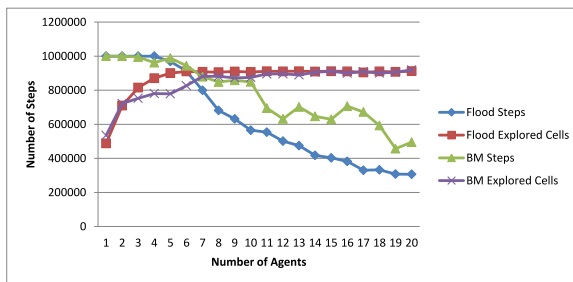


Fig. 7. Results Street of houses, 1000×1000 cells

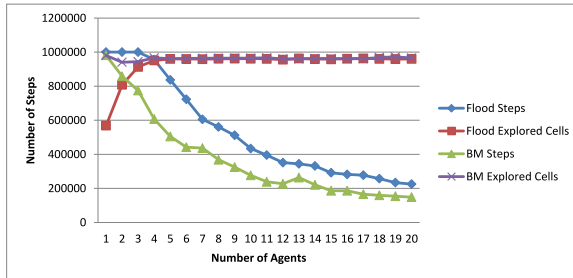


Fig. 8. Results Plain, 1000×1000 cells

All figures from Figure 4 to Figure 8 show, that the run time of the *Multi-Agent Flood* algorithm is a little bit slower than the *Brick and Mortar* algorithm. The diagrams depicting the results of the bigger test cases (Fig. 7 and 8) show, that this difference is smaller in the larger scenarios, especially in Fig. 7, where the MAF algorithm is faster than the BM algorithm. This is particularly interesting, as the movement routine of the MAF agents is still unoptimized and each agent will return to the base to report the location of the latest point of interest, that was found by said agent. The agents of the BM algorithm will continue their search and, as mentioned above, will try to avoid already visited cells. This also reaffirms the results of [4] that the loop detection routine is rather expensive.

The results of the different sized scenarios also show, that the MAF algorithm scales rather well in the number of agents and the size of the map. As mentioned above, the algorithm needs at least about five agents to compute results with a satisfying coverage of 95% of the scenario and search hit rate of 95%. These results are independent of the underlying map size, whether the map is built of 500×500 cells or 1000×1000 cells.

7 Conclusion

In this paper we proposed a new sort of search and rescue algorithm, that combines two phases into one allowing the rescue teams to act earlier on the information and work in parallel with the search robots. This new approach ignores the lack of wireless communication and adds the possibility to be able to do maintenance work on the robots at their return to the base.

The results of our simulations show, that the additional capabilities do not really impair the run time of the search phase. This is especially true in bigger scenarios like streets of houses or big buildings, where the “naive” version of the MAF algorithm can be faster than the benchmark algorithm. Future work will be aimed at optimizing the current movement and monitoring methods to shorten the run time of the algorithm. Additional possibilities exists in the application of this algorithm in dynamic scenarios, for example scenarios, that allow the change of the environment during run time or the movement of the victims as long as they are not found by an agent.

Depending on the scenarios our algorithm can even be extended with the addition of wireless sensors or even simple line of sight communication, which would allow the agents to communicate directly between them. Enabling the exchange of data about already explored terrain and already uncovered paths back to the starting point. The “Return Mode” can also be optimized to yield shorter paths on the return to the base.

References

1. Bellman, R.: *Dynamic Programming*. Princeton University Press, Princeton (1972)
2. Birk, A., Pathak, K., Schwertfeger, S., Chonnaparamutt, W.: The IUB Rugbot: an intelligent, rugged mobile robot for search and rescue operations. In: *IEEE International Workshop on Safety, Security, and Rescue Robotics (SSRR)*. IEEE Press (2006)
3. Dorigo, M., Gambardella, L.: Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation* 1(1), 53–66 (1997)
4. Ferranti, E., Trigoni, N., Levene, M.: Brick & Mortar: an on-line multi-agent exploration algorithm. In: *IEEE International Conference on Robotics and Automation*, pp. 761–767. IEEE (2007)
5. Ferranti, E., Trigoni, N., Levene, M.: Rapid exploration of unknown areas through dynamic deployment of mobile and stationary sensor nodes. *Autonomous Agents and Multi-Agent Systems* 19(2), 210–243 (2009)
6. Murphy, R.R., Tadokoro, S., Nardi, D., Jacoff, A., Fiorini, P., Choset, H., Erkmén, A.M.: *Search and rescue robotics*. In: *Handbook of Robotics*, pp. 1151–1173. Springer (2008)
7. Nagatani, K., Kiribayashi, S., Okada, Y., Tadokoro, S., Nishimura, T., Yoshida, T., Koyanagi, E., Hada, Y.: Redesign of rescue mobile robot Quince. In: *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 13–18. IEEE (2011)
8. Ranjbar-Sahraei, B., Weiss, G., Nakisae, A.: A multi-robot coverage approach based on stigmergic communication. In: Timm, I.J., Guttmann, C. (eds.) *MATES 2012*. LNCS, vol. 7598, pp. 126–138. Springer, Heidelberg (2012)
9. Ruangpayoongsak, N., Roth, H., Chudoba, J.: Mobile robots for search and rescue. In: *IEEE International Safety, Security and Rescue Robotics, Workshop*, pp. 212–217. IEEE (2005)
10. Sakakibara, T., Kurabayashi, D., et al.: Artificial pheromone system using RFID for navigation of autonomous robots. *Journal of Bionic Engineering* 4(4), 245–253 (2007)
11. Zhu, Q., Wang, L.: A new algorithm for robot path planning based on scout ant cooperation. In: *ICNC 2008, Fourth International Conference on Natural Computation*, vol. 7, pp. 444–449. IEEE (2008)

A Practical Security Infrastructure for Distributed Agent Applications

Lars Braubach, Kai Jander, and Alexander Pokahr

Distributed Systems and Information Systems
Computer Science Department, University of Hamburg
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
{braubach,jander,pokahr}@informatik.uni-hamburg.de

Abstract. Security is a vital feature for most real-world distributed applications. For applications using an agent-based middleware, comprehensive support for many security-related aspects can and should already be provided by the platform in a holistic manner. Moreover, security in practice does not only concern the enforcement of well-known security objectives like authenticity and confidentiality but also requires a simple yet effective usage concept that renders it easy for developers to define their security requirements. In this paper a security concept and implementation for multi-agent systems is introduced, which focuses on external, i.e. inter-platform, security aspects. The solution encompasses a usage concept distinguishing security intents from realization details and allows service providers as well as service clients to impose security constraints on communication relationships. The security concept requirements have been elicited from and the results have been evaluated with a real-world company project in the area of distributed workflow management in business intelligence.

1 Introduction

Distributed systems allow applications to be used in a widely distributed manner, which confers unique advantages over more centralized or server-based approaches such as increased performance and fault-tolerance. These systems offer application platforms, which are capable of executing parts of the software and enabling the communication between them and other platforms in a convenient manner for the developer. However, a large number of real-world applications especially in sectors such as banking, health and communication are mandated, sometimes by law, to fulfill certain requirements regarding the security of the system. Such requirements are frequently relegated to an afterthought, for example by encrypting traffic using HTTPS, which often fails to address application requirements such as fine-grained access control. In some cases, security requirements are ignored by the software platform and thus deferred to the application level, which means that every application is forced to reimplement its own security measures. Given that initial implementations often have flaws, this amplifies the problem by given each application its own chance to include the same flaws and requiring separate modification to address the problem.

As a result, it would be beneficial for the platform itself to offer a useful set of security features that can be employed by all applications being executed on the platform. While some features are too specialized to be provided in a generalized manner, a number of features can be offered which are often not included in such platforms. The first step towards implementing security features in a platform is to ensure that basic security is maintained even in an only partially controlled environment that is used by multiple stakeholders. Specifically, this means that platforms should *restrict access* to functionality by default and only allow it when specified. This allows the platform to prevent unauthorized access and achieve specific security goals[11]. In addition to the access restriction, some applications also require that *non-repudiation* can be established. This means that actions taken on a platform by an entity can be accounted for and it can be proven that specific service requests were placed by specific issuers.

This first step allows the inclusion of security features from the perspective of a user of the MAS infrastructure, such as an application developer. The first of these user security objectives is the *integrity* of communication between platforms. This means ensuring that an attacker will be unable to tamper with messages without the target platform noticing the change. If the communication is performed using a public network, the issue of eavesdropping needs to be accounted for by establishing *confidentiality* of the data exchanged between platforms. Since platforms restrict access by default and allow only some entities access to its functionality, the *authenticity* of the entity needs to be established. After authentication has been established, the *authorization* of the entity to perform a function needs to be verified.

In addition to the user security objectives, the usability of the security model provided by the platform must be part of the overall security concept of the system and balanced against the security requirements[12]. Here, two groups are particularly important. First, application developers should be offered easy-to-use APIs and configuration options to integrate the offered features in their application. Second, administrators of nodes that run the platform must be offered management tools, for example to generate and distribute certificates.

This paper proposes a platform-based security model for agent applications that is motivated by a real-world usage scenario which will be introduced in the next section. In Section 3, related approaches will be presented. The security concept for the platform-based approach will be introduced in Section 4, followed by an elaboration of the implementation of the approach in the Jadex agent platform in Section 5. The approach is then evaluated in Section 6 based on the real-world scenario, followed by a conclusion in Section 7.

2 Real World Scenario

In a commercial project called DiMaProFi (Distributed Management of Processes and Files) carried out with the company Uniiue AG¹ a new business

¹ <http://www.uniiue.de/>

intelligence tool is currently under development. The tool will target the modeling and automated execution of distributed ETL (extract, transform, load) workflows. These kinds of workflows serve the purpose of collecting and pre-processing data and files and finally move them into a data warehouse, which is subsequently used by domain experts for sales and other business evaluation tasks. The targeted scenario is naturally distributed, spanning often more than one network as the relevant source data is generated at different customer sites.

In such a setting many security related problems arise:

- The DiMaProFi application is distributed and needs to work on different nodes, on the one hand to bring about its functionality and on the other hand to save time by distributing the system load. This means that the nodes of the system have to form an overlay network which permits access to the members but prohibits access from other nodes and protects the integrity and confidentiality of the exchanged messages.
- Another important aspect is that the system needs to have access to password protected resources like databases and also the target data warehouse. To avoid the distribution of passwords among all participating nodes, a mechanism for acquiring authentication at runtime needs to be provided. In DiMaProFi, a specific password safe is used which can be interrogated by authenticated parties to obtain credentials for access restricted resources. For this approach it is important to support origin authentication of the different parties at the password safe and also provide confidential messaging to protect credentials from being eavesdropped.
- Regarding the communication with the customers it is important that in error cases the activities of the system have been exactly monitored and recorded. On the one hand this facilitates the failure recovery and on the other hand it might be important with respect to legal responsibilities concerning the service level agreements with the customer. For this purpose the correctness of the monitoring data is significant and should be safeguarded by a non-repudiation mechanism.

From this description it becomes obvious that in distributed applications like DiMaProFi security objectives become vital and mechanisms for access restriction, authentication, integrity and non-repudiation need to be established. These objectives are also considered in literature to be of primary importance [7] for distributed system security.² Moreover, these security solutions should be practical in the sense that they are easy to install, configure and maintain, i.e. security should not become a complicated overhead possibly fostering the bypass of important mechanisms.

3 Related Work

While security features are a frequent requirement for applications, security aspects are often either overlooked or only superficially considered when designing

² We decided to exclude availability here as it is more related to system design as a whole in contrast to basic design entities like agents.

multi-agent systems [8]. Most platform-based systems rely on the use of standardized or established algorithms such as AES [10] and RSA [16] and protocols like SSL/TLS [3] and Kerberos [9] to meet the requirements of the applications. Since designing both security algorithms and protocols always carries the risk of flaws and established approaches have been used and vetted for many years, using established solutions is generally a good idea. However, the approaches often lack good integration with the platform, resulting in a system that is difficult to configure and use.

Regarding distributed systems, one can distinguish two classes of security: Internal security, which attempts to enforce restrictions between components such as agents executing on the same platform, and external security, which centers on the communication between different platforms and the enforcement of restrictions between platforms. Generally speaking, internal security is fairly hard to enforce since code executed within the same process instance is usually able to circumvent security measures, either through approaches like reflection APIs or by simply reading or manipulating the process memory. A notable exception are sandboxing approaches [14], which attempt to encase certain parts of locally executed code by restricting its API access and memory reading capabilities in order to contain the sandboxed part of the application within a known scope of permissions.

A prominent example of this approach is the Java Security Manager [5], which allows the application of method-based restrictions on parts of the executing code. However, since this approach relies on complex code analysis and interpreter assistance, it is prone to include subtle flaws [4]. As a result, internal security remains difficult to maintain. Furthermore, basing security policies on the method level is often too fine-grained to be easy to configure and often allows attacks if not done with extreme caution. This complexity can be considerably reduced when considering external security.

Nevertheless, some agent-based approaches attempt to enforce internal security in addition to external security. For example, the JADE-S add-on [17] for the JADE agent platform allows the user to restrict access to platform services using access control lists (ACLs) and user authentication. While this addresses one kind of authentication with regard to administrative aspects of platform management, there is no mechanism to assist authentication for agent interactions and services. JADE-S also offers the option to encrypt and sign messages between agents, potentially opening the possibility of agent authentication and message confidentiality. However, this is supported by key pairs held by each agent. This means that enforcing inter-agent secrecy on the same platform requires the enforcement of internal security with the aforementioned complexity and there appears to be no readily available function for key distribution, which greatly increases administrative overhead and reduces usability. This problem is magnified by the lack of scalability because each requires a key pair.

Other agent systems attempt to address the problem of keeping confidentiality between platform and agent or between multiple agents. The concerns here are less about communication but rather the danger of compromising the

execution space of either the agent or the platform. For example, in [6] a method is proposed which lets mobile agents perform computations on encrypted data, preventing the platform from gaining access. In the opposite direction, sandboxing approaches are once again chosen to maintain internal security of the platform with regard to the mobile agent. Similarly, the use of controlled query evaluation (CQE) techniques is proposed in [1] in order to prevent agents from revealing confidential information during interactions. These approaches focus primarily on maintaining the confidentiality either within the same MAS application or between executing platform and agent while the approach presented in this paper focuses on maintaining the confidentiality of inter-agent communication against an outside attacker.

Due to the difficulties associated with the enforcement of internal security through sandboxing, many approaches forego this approach, instead focusing on the easier-enforced external security similar to the approach presented in this paper, which relies either on well-tested operation system process security, virtualization or physical separation of machines to aid in access restriction. Since additional platforms can always be added automatically as additional processes, this reduces the implementation difficulties of sandboxing by offloading the issue to the operation system process security, which receives a higher degree of scrutiny. For example, WS-Security provides features for encryption and signatures which can be used to secure web services [13]. This approach provides support for external security for web service calls. However, while aspects such as certificate formats and the use of security tokens is part of this approach, it does not offer an easy key distribution mechanism. Furthermore, authentication relies, in large part, on the application layer with support mostly provided for basic aspects such as confirmation of signatures.

4 Security Concept

In this section, the security concept is introduced. First, the scope of the security concept is explained, i.e., the setting in which security measures are applied and which security objectives are supported. Afterwards details of the security model are illustrated to show how the security objectives can be achieved.

4.1 Scope of the Security Model

Multi-agent systems can be deployed in various settings ranging from simple closed networks to internet scale distributed systems. The security model presented here aims at open distributed systems, in which agents are used to realize some of the systems functionality. This functionality is exposed to the outside as services, which e.g. may be accessed using message-based interaction protocols. It is assumed that agent platforms represent the nodes in the network and each agent platform is under some administrative control.

The focus is on security issues due to agent services being accessed from the outside (cf. Fig.1). With regard to the usage of these services four common security objectives can be identified.

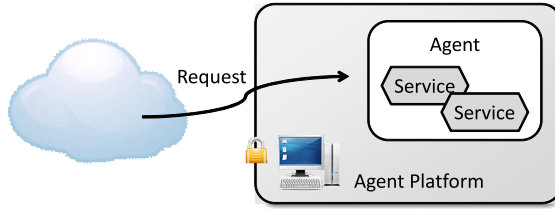


Fig. 1. Basic security model

Access Restriction: The functionality must only be provided to those who are allowed to use it.

Authentication: The agent should be able to uniquely identify the user of its functionality, e.g. for accounting purposes.

Confidentiality The usage of the functionality as well as private data should not be visible to others.

Non-Repudiation Users of the functionality should not be able to deny that a certain request was used.

Due to the focus on outside requests, certain security issues are intentionally not covered in the model. First, issues of mobile code are excluded, because mobile agents are considered unnecessary for many real world applications. Moreover, security issues due to an attacker having direct access to the computer running the agent platform are not considered. Such an attacker might be able to install malicious code or sniff sensitive data from local memory. Yet, these kinds of attacks do not require agent-specific solutions, but can be already covered with user-based access control, anti-malware-checkers, etc.

4.2 Security Model Details

Security objectives are related to individual services. Each service may require a different combination of security objectives. E.g. instantiating a new data management process in DiMaProFi only requires access control but no confidentiality, because no sensitive data is transferred. In a later step of the process, confidentiality might become necessary, e.g. when the process accesses a sensitive customer file.

Therefore, security policies are introduced to allow fine-grained, yet easy to use configuration of security objectives. These security policies may be specified both at the sender as well as the receiver side (cf. Fig. 2). Typically, the receiver side providing the service will already specify the general security objectives that apply to any usage of the service. Therefore, the receiver can choose to demand that all access to the service must be authorized, that data must always be transmitted confidentially etc. Furthermore, the sender side can supply a custom policy for each request to demand further security measures. For example, if a service does not demand confidentiality in general, the sender can still request that communication be encrypted for a specific interaction.

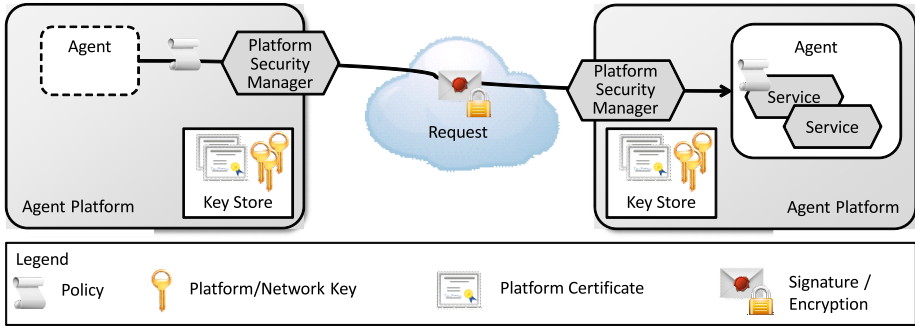


Fig. 2. Security model details

To achieve the desired security objectives, a security manager is introduced that operates as part of each agent platform. This approach allows all security-related functionality to be handled by the agent platform itself and relieves agent programmers from tedious and time-consuming implementations of security details. The platform security managers are responsible for processing the requests before they are sent and after they are received. Each request between two agents is thus routed through two platform security managers, one at the sender and one at the receiver side (cf. Fig. 2). In the following, for each security objective it will be described, how it can be realized inside the platform security managers using further security concepts, such as keys and certificates.

Access Restriction. By specifying an according security policy for an agent service, a developer can control, if agents from other platforms may access the service. For simplicity, the model allows two modes of access restriction. One that requires authentication and one that does not. The simple mode without authentication relies on the concept of trusted platforms and trusted networks as described below. The idea is that an agent platform should allow access to all services of its agents when requested by remote agents from trusted platforms or from platforms in trusted networks. All other agents are only allowed to access services, which have been explicitly marked as public by the developer. The validation of trusted platforms is done by using platform and network keys, which are kept in a key store on each platform. The difference between platform and network keys is that a platform key allows only access to a single platform, while a network key allows access to a logical network of platforms. A platform only has one own platform key but it may participate in any number of trusted networks and thus may have multiple network keys. In addition to its own platform key, the key store also contains the known keys of remote platforms.

An example is shown in Fig. 3. Platform *A* has a local platform key as well as network keys for networks *N1* and *N2*. As it shares network and key for *N1* with platform *B*, platform *A* will consider *B* as trusted. Similarly, platform *E* is trusted by *A* as both share the key of network *N2*. Furthermore, platform *G*

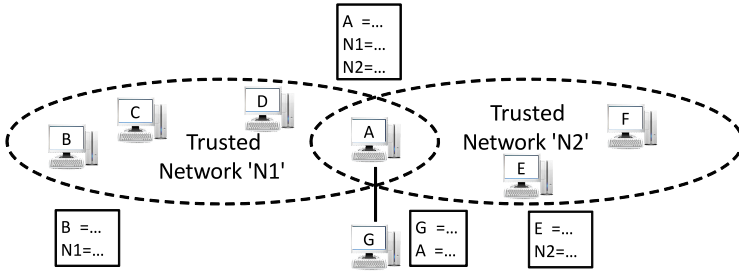


Fig. 3. Trusted platforms and trusted networks

is trusted by A , because G is in possession of A 's platform key. Due to their disjoint network memberships, platforms B and E would not trust each other.

Whenever a request is issued to a remote platform, the security manager on the sending side will intercept the request and enhance it with digests of the relevant platform and network keys. The security manager at the receiver side also produces digests of its local keys and checks if one of them matches the sent digests. If a match is found, the platform can be considered as trusted as it shares a common secret with the sender. Due to the use of digests, no keys need to be transferred and therefore cannot be sniffed by eavesdroppers. Furthermore, using timestamps as part of the digested text provides protection against replay-attacks.

The second access restriction mode with authentication allows more fine-grained control over which services may be accessed by which remote platform. This can be specified in two ways. First, the names of allowed platforms can be annotated in the service policy. Second, virtual group names can be specified that are later mapped to concrete platform names. The first way is simpler, but may introduce maintainability issues as changes in platform names may require security policies to be updated. The second way allows for role-based access control as the virtual names can be used to represent roles. Therefore, developers can specify which roles are allowed to access which services and administrators later configure, which platforms are allowed to play which roles. The details of the authentication mechanism are described next.

Authenticity and Integrity. Authenticity is about establishing the identity of a communication partner and integrity means that no tampering with message contents is possible. They come hand in hand as requirements for other security objectives such as access control and non-repudiation. In the presented security model, authenticity and integrity are established by the use of digital signatures and platform certificates. When authentication is required, the security manager of the sending platform will add a digital signature to the message, which is produced by the private key of the platform. To validate the authenticity and integrity using the digital signature, the security manager of the receiving platform uses the platform certificate, which contains the public key of that platform.

The means that only when the receiving platform already has the certificate of the remote platform, the digital signature can be validated. Therefore an important aspect of authentication is the distribution of platform certificates.

A common and safe but tedious way is to manually install the certificates. In a network of n platforms, $n * (n - 1)$ installation operations would be required to be performed by platform administrators. Therefore alternative solutions are provided to simplify the process. These solutions allow obtaining certificates during the process of validation a signature and thus are called certificate acquisition protocols. The first protocol uses a certificate server as a trusted third party. Due to the central server only $n * 2$ operations would be required to install each platform certificate at the server and also install the server certificate at each platform. An additional advantage of the approach is that when adding a platform, only the newly added platform and the server need to be considered as no additional certificates need to be installed at previously existing platforms.

The second protocol is based on trust relationships to other platforms. It realizes a consensus mechanism that asks a set of available platforms for a specific and currently unknown platform certificate and compares the received results. If a configurable threshold of platforms has delivered the same certificate it is accepted and added to the internal key store. The number of manual installation operations of this protocol directly depends of the chosen threshold for accepting a certificate. In the best case (threshold=1) 0 manual installations are necessary, while in general with (threshold= x) $(x - 1) * n$ operations are required.

In order to reduce the efforts of manual installation to a minimum a two-staged process can be used. In the first stage the network of platforms, e.g. running in a company intranet, has to be cut-off from the internet so that so no malicious platforms can participate. In this initialization phase the certificate acquisition has to be enabled at the security manager of each platform and e.g. the consensus protocol can be used with threshold=1. After having set up the network a complete certificate exchange is started in which each platform requests certificates of all other visible platforms. After the exchange has finished the initialization phase is completed, the certificate acquisition can be turned off and the internet connection can be reestablished.

Confidentiality. Confidentiality can be specified at the receiver side for all requests to a service as well as at the sender side for a specific request. In both cases, confidentiality means that all communications pertaining to the service usage (i.e. requests as well as replies) should be encrypted to prohibit outsiders to gain access to the exchanged information. When performing confidential communication, the platform security managers are responsible for ensuring that the corresponding messages are sent through secure channels. For confidential communication between different platforms, the use of secure message transport protocols is obligatory. If no such transport is available, the sending agent is notified about the request failure and the message is discarded.

Non-repudiation. Non-repudiation means that it can be proven that a certain request was issued by a certain party. To achieve this it is required that

authenticity and integrity of requests can be established. To prove the issuance of a request at a later point means further that all relevant requests should be logged. This is achieved using a mechanism for monitoring the service invocations. For each agent it can be specified, which services are of interest and thus which requests need to be written to the log.

4.3 Usability Concerns

One main focus of the presented security model is ease of use at two levels - for the agent programmer and for the platform administrator. The presented model is easy to use for agent programmers, because security issues are specified as non-functional properties. As a result, the agent implementations do not need to deal with security issues leading to an advantageous separation of concerns. Furthermore, using the non-functional properties, security configurations can easily be added to existing agent implementations. Usability is further facilitated by the flexibility and simplicity of the model. For example, for access control, the developer can choose from three access modes: public, trusted, and authenticated access. The public access mode allows easily enabling global access to non-critical public services without compromising security of critical services. The trusted access mode is enabled by default thus automatically protecting any service implementation without further effort from unauthorized access. Therefore, agent platforms and applications can safely be hosted in open networks without having to consider security at first. Finally, authenticated access allows fine-grained role-based access control, if required.

For administrators, usability issues arise as the platforms need to be configured with two conflicting goals in mind: 1) keep the platform as closed as possible to prevent any malicious activities and 2) keep the platform as open as necessary for the distributed application to operate correctly. The concept of trusted networks provides a simple solution for this conflict that is applicable to many real world situations. By establishing a trusted network, an administrator can easily allow platforms from heterogeneous company networks to interoperate without exposing their functionality to the outside. More control about which platforms are allowed to access which services can be exercised by using platform certificates, which come at the cost of some administrative overhead for managing local and remote certificates on each platform. To ease the administrative burden of certificate management, the model proposes protocols for (semi-)automatic certificate exchange.

5 Implementation Aspects

The proposed security concept has been implemented in the Jadex platform [15]. The platform uses an extended agent concept called active components, which in essence allows agents to expose and use explicit services with asynchronous object-oriented interfaces. Regarding the usage of security aspects, a distinction is made between the security objectives and their enforcement, which is the task

```

@SecureTransmission
@Authenticated({"DatabaseUser", "Admin"})
public interface IPasswordSafeService {
    public IFuture<Credentials> fetchCredentials(String resourceid);
    ...
}

```

Fig. 4. Provider side security usage example

```

IFileRegistrationService service = searchService();
ServiceCall call = ServiceCall.getInvocation();
call.setProperty(SecureTransmission.SECURE_TRANSMISSION, true);
service.registerFile(file);

```

Fig. 5. Client side security usage example

of the platform’s security manager realized as security service. The specification of security objectives differs for service provider and client side.

5.1 Provider Side Usage

The specification of aspects at the provider side is handled with different Java annotations that can be attached to a service method or the service interface itself (which means that the objective is applicable for all methods of the interface). In Fig. 4 it is highlighted with an example how security annotations can be used. In this case a cutout of the DiMaProFi password safe service is shown, which is besides other things in charge of providing credentials for agents that need to access password restricted resources like databases. In order to make all methods confidential and authenticated the corresponding annotations (`@SecureTransmission` and `@Authenticated`) are attached to the interface `IPasswordSafeService`. Additionally, the authentication is parameterized with two roles (“DatabaseUser” and “Admin”), which are allowed to access the password safe. These role names are mapped to concrete platform names in the security manager of the platform.

5.2 Client Side Usage

At the client side security objectives can also be requested dynamically.³ This is achieved by a concept similar to a thread local⁴ but not based on a thread but on a service invocation, i.e. a specific service call object allows for equipping an invocation with additional meta data that is automatically preserved between caller and callee during the complete call, regardless if the call is performed locally or remotely. The service call object can be fetched and attributed with security objectives by the caller before a service call is issued. An example usage, again taken from DiMaProFi, is shown in Fig. 5. It can be seen, how a service call can be made confidential at runtime. For this purpose the service call object is

³ Please note that not all security objectives can be set up without the receiver side, e.g. authentication requires the receiver side to declare which identities are allowed to use a service.

⁴ A thread local is an object that belongs to a thread and which can be used to store and retrieve data without having to pass it explicitly as method parameter value.

fetches via the static method `getInvocation()` of the `ServiceCall` class. Afterwards, the secure transmission property is set to true in the call object and the service call (here a file registration operation) is performed as usual.

5.3 Administration Tools

To facilitate the administration of security aspects of the platform tool support is provided. In Fig. 6 a screenshot of the platform security manager interface is depicted. The tool supports four different use cases via a tabbed view. First, the local password settings can be configured, i.e. the platform password can be changed, en- or disabled. Second, certificate and key pair administration can be accessed via the key store tab (active in Fig. 6). This view displays the content of the currently used key store, i.e. the contained key pairs and certificates and on the other hand it allows for manually adding or removing keys and certificates of trusted platforms. Importing a certificate can be either done by using a certificate file or by directly requesting it from the corresponding target platform (if that platform is currently online). Moreover, the view allows for selecting and configuring the used certificate acquisition mechanism (lower right of Fig. 6). It is also possible to completely disable automatic certificate acquisition. Third, the remote passwords tab can be used to view, add or remove credentials of known remote platforms. Alternatively, in the fourth tab network names and passwords can be managed to set up virtual platform networks without having to specify individual platform passwords.

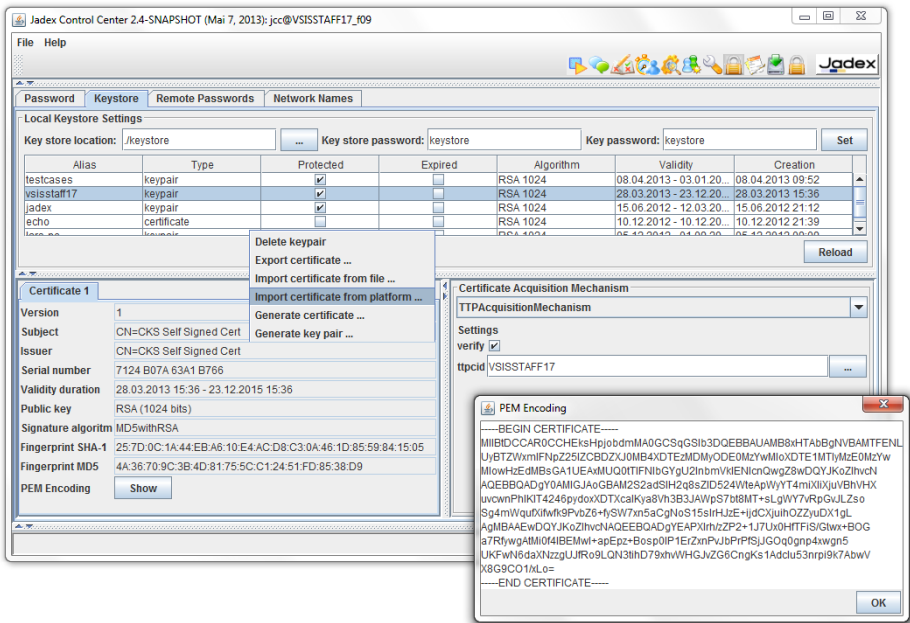


Fig. 6. Platform security manager user interface

6 Evaluation in Practice

The evaluation is based on the real world application scenario presented in Section 2. The DiMaProFi system consists of loosely coupled nodes, each hosting a Jadex platform and application components. The application components use local and remote services of each other in a transparent way. Service discovery is performed dynamically at runtime and can take into account non-functional properties. The overlay network of agent platforms is created by an awareness mechanism, which automatically discovers other nodes [2].

- To prevent unauthorized platforms from using services of DiMaProFi the virtual network access restriction mechanism is employed. The usage is very simple as it is sufficient to start each platform with an application specific virtual network name and password. In this way these platforms share a common secret and can communicate seamlessly. To further avoid denial of service attacks the visibility of the platforms is also hidden. This means that the platforms use a private relay server for managing awareness between different physical networks.
- In order to solve the problem of access to protected resources the password safe concept has been realized using a corresponding service that allows for requesting credentials for a protected action. The service itself exposes its methods only for authenticated users by declaring a set of allowed user names. Within the platform security manager these user names are mapped to trusted platform certificates. Incoming requests need to provide a signed digest that is verified before the call is processed.
- The monitoring infrastructure of DiMaProFi is based on the service and component monitoring of the Jadex platform. This infrastructure creates events for service calls in the system and automatically sends them to a local monitoring service. This service saves the events and offers a subscription based interface for fetching possibly filtered events. For DiMaProFi a custom component has been realized that uses the monitoring service and employs workflow specific rules to detect errors as early as possible. Given that authentication is used in service calls the monitoring logs can be used for proving also non-repudiation of the corresponding invocations.

Besides achieving the overall security objectives within the project it was of crucial importance to tailor the solutions in a way that they become easy to administer and use. In this respect especially the virtual security network and the certificate distribution concepts were considered very helpful by our practice partners as those render it possible to change the underlying nodes infrastructure without huge configuration efforts.

7 Conclusion

In this paper an approach for achieving external security within multi-agent systems operating in open networks has been presented that is directly motivated by

real world requirements of a company project dealing with business intelligence workflows. The security concept supports the achievement of the security objectives integrity, confidentiality, authentication and non-repudiation for service calls. In contrast to most other works, usability was a key factor of the proposed solution. Hence, the usage as well as the administration have been designed to be as simple as possible with usage based on security annotations at the service provider side and dynamically added security meta information at the client side. Moreover, tedious aspects of administration have been resolved at the platform level. Using virtual networks makes it easy to set up large sets of communicating nodes without configuration efforts. Furthermore, also automatic platform certificate acquisition protocols have been included, which largely relieve an administrator from installing platform certificates manually. As part of future work it is planned to include security aspects also in the service search mechanism. This will allow searching for services satisfying specific security features.

References

1. Biskup, J., Kern-Isberner, G., Thimm, M.: Towards enforcement of confidentiality in agent interactions. In: Proceedings of the 12th International Workshop on Non-Monotonic Reasoning (NMR 2008), University of New South Wales, Technical Report No. UNSW-CSE-TR-0819, pp. 104–112 (September 2008)
2. Braubach, L., Pokahr, A.: Developing Distributed Systems with Active Components and Jadex. *Scalable Computing: Practice and Experience* 13(2), 3–24 (2012)
3. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. Internet Engineering Task Force (August 2008)
4. Goichon, F., Salagnac, G., Frénot, S.: Exploiting Java Code Interactions. Technical Report RT-0419, INRIA (December 2011)
5. Gosling, J., Joy, B., Steele, G., Bracha, G., Buckley, A.: The Java Language Specification, 7th edn. Addison-Wesley Professional, California (2012)
6. Lee, H., Alves-Foss, J., Harrison, S.: The use of encrypted functions for mobile agent security. In: HICSS, pages 10. IEEE, New York (2004)
7. Moffett, J.D.: Distributed Systems Security. A. Kent, J.G. Williams 15 (1995)
8. Nagaraj, S.V.: Securing multi-agent systems: A survey. In: Meghanathan, N., Nagamalai, D., Chaki, N. (eds.) *Advances in Computing & Inform. Technology*. AISC, vol. 176, pp. 23–30. Springer, Heidelberg (2012)
9. Neuman, B.C., Ts'o, T.: Kerberos: An authentication service for computer networks. *Comm. Mag.* 32(9), 33–38 (1994)
10. NIST. Advanced Encryption Standard (AES) (FIPS PUB 197). National Institute of Standards and Technology (November 2001)
11. NIST. Underlying Technical Models for Information Technology Security. National Institute of Standards and Technology (December 2001)
12. Norman, D.A.: The way i see it: When security gets in the way. *Interactions* 16(6), 60–63 (2009)
13. OASIS. Web Services Security: SOAP Message Security 1.1 (February 2006)
14. Oey, M., Warnier, M., Brazier, F.: Security in Large-Scale Open Distributed Multi-Agent Systems. In: *Autonomous Agents, Rijeka, Croatia*, pp. 1–27. IN-TECH (2010)

15. Pokahr, A., Braubach, L.: The active components approach for distributed systems development. *International Journal of Parallel, Emergent and Distributed Systems* 28(4), 321–369 (2013)
16. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 21, 120–126 (1978)
17. Vila, X., Schuster, A., Riera, A.: Security for a Multi-Agent System based on JADE. *Computers & Security* 26(5), 391–400 (2007)

Verifying MSMAS Model Using SCIFF

Emad Eldeen Elakehal¹, Marco Montali², and Julian Padget¹

¹ Department of Computer Science
University of Bath, BATH BA2 7AY, UK
emad@itu.dk, jap@cs.bath.ac.uk

² KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano, Italy
montali@inf.unibz.it

Abstract. MSMAS is a software development methodology that facilitates the design and development of complex distributed systems based on the multiagent systems paradigm. MSMAS explicitly supports the institutional organisational structure and follows a declarative modelling style to specify behavioural restrictions on the members of the institution, their roles, the business processes regulating their behavior and the communication protocols regulating their mutual interactions. All these aspects are visually represented, by adapting the DECLARE graphical language, proposed for the declarative specification of constraint-based business processes. In this paper we discuss the main elements of MSMAS, and show how they can be equipped with a formal, expectation-based semantics, tailored to the SCIFF Abductive Logic Programming-based framework. In particular, we show how the MSMAS constructs can be formalized in SCIFF, and then exploit this correspondence to specify and verify formal properties over MSMAS models, by leveraging on the SCIFF reasoning capabilities.

1 Background

With a growing interest in modelling the social structure of modern distributed systems and increased research activities around using role norms and institutions as an organisational type to capture these social aspects, it seems that modelling efforts are disconnected from implementation at the application level. Some metamodels lack supporting design tools or if not they lack design verification and run-time validation capabilities. The MSMAS [6,5] methodology aims to establish a link between modelling and implementation by combining business oriented metamodeling with institution and role modelling and supports a formal proof mechanism for design and runtime validation. MSMAS allows multi agent system (MAS) designers to model self-managing MASs using visual graphic models. Here, we take self-managing to mean the ability of the system to recognize execution errors or undesired behaviour and the ability to respond by replanning in order to recover from the failure or to stop the undesired activities. MSMAS has three phases: the first phase is to capture the system requirements through the *Use Cases Models* and to create the *System Goals Model*. The second phase starts with the high level design of the required *Business Processes* to achieve the system goals and the specification of the system organisational structure through the *Institutions Models*. Then, a detailed design of the business activities, a full specification

of the *System Participants* and the specification of *Communication Protocols* that defines how system participants can coordinate their activities, interact with one another and exchange information. The third phase concerns implementation, where the user can export the system specification in either of the two available formats. The first is the SCIFF formal framework [1], which supports the designer in the assessment of the produced model, checking its correctness and verifying whether it meets desired properties, also taking into account possible execution traces produced by the system. The second format is RDF that offers a basis for transformation to any other execution languages such as JADE, JASON, etc. Or can be mapped to another RDFs such as the frame work proposed by Alberola et al[7]. Verifying SCIFF model is good indicator of the correctness of the RDF model, because both models are reflecting the same MSMAS metamodel.

In MSMAS, the system designer can set constraints on the business processes and/or their activities, as well as on the system participant roles. Any activity without a constraint can be executed an arbitrary number of times in an arbitrary order as long as its preconditions are satisfied, while the constraints on system participants' roles are used to specify the accepted behavioural patterns. To impose dynamic constraints on the activity execution, MSMAS uses the graphical notation of DECLARE (see <http://www.win.tue.nl/declare/>, retrieved 20130627), deriving from the DecSerFlow/ConDec languages developed by van der Aalst and Pesic [14,13,11]. DECLARE is a declarative language for modelling and enacting constraint-based business processes. We chose a declarative approach for MSMAS because it is well-suited to the dynamic nature of MASSs, and because DECLARE offers a simple graphical notation with a powerful and flexible formal representation.

DECLARE takes an open approach where the relationship constraints that are set between two or more activities can be either positive and negative. Positive relationships are normally used to state that a certain activity is expected to occur when a certain system state is achieved, while negative relationships state forbid the execution of activities when a given state of affairs holds. DECLARE offers a number of loosely-coupled template relations, which go beyond the standard sequential relationships of classical process specification languages. An example is the *responded presence* constraint, which states that if the source activity A is executed, then the target activity B must be executed as well, either before or after the execution of activity A.

Many formal models for agent-based systems have been proposed and they present useful approaches for building such systems. We believe MSMAS makes a valuable contribution to this line of research because of its ability to model scenarios, in which the system components and human participants interact governed by social norms. We consider this is an important aspect, as modelling only individual agent aspects cannot cover all the issues that affect how they interact and coordinate their behaviour to allow the system to achieve its goals; hence considering the social aspects of these individual agents becomes necessary. Incorporating social and organizational structure complicates the MAS model, however by following the MSMAS methodology and breaking down the system into smaller organisations and encoding the different behaviour patterns into roles, we argue that complication is contained. Furthermore, allowing the system properties to be assessed during design time with the support of a formal

framework helps in modelling societally-structured systems and the use of a declarative style enables monitoring and runtime verification [4,1,10].

2 Formal Model of MSMAS

Formal modelling methods of software comprise two activities: *formal specification* and *verification*. Formal specification permits the deployment of an accurate specification of the system behaviour that allows for a precise modelling of the system, while verification aims at proving that the model of the system complies with the intended requirements, and meets the desired properties.

Our evaluation of the literature and existing and past approaches, leads us to the conclusion that MSMAS can be based on just four concepts that are sufficient for a MAS description to be able to provide answers to the following four questions:

1. What is the purpose of building the system and its individual components? This question is answered by defining **System Goals** (SG) as the first core concept.
2. How can the system achieve its goals? The answer lies within the second core concept which is the system **Business Processes** (BP) and their activities.
3. Who or what is responsible for the execution of each business process activity? This is answered by the third core concept of **System Participants** (SP).
4. Through which organisational structures do the system participants interact and what roles can they play? This is answered by defining **System Institutions** (SI).

So, in our approach, a MAS is a 4-tuple: $MAS_{msmas} = \langle SG, BP, SP, SI \rangle$ and hence, in order to connect these four concepts, we address the modelling of: (i) System Participant Institutional Roles, (ii) System Participant Communication Protocols, and (iii) Business Processes Relationships. The system goals are the main drivers of the business processes and all activities in MSMAS are goal-directed so the formalisation above covers the complete MSMAS system view. In the rest of this section we go into more detail about each of the above concepts.

Institutional Roles. MSMAS requires the explicit statement of the society's organisational structure, where the system is organised in a number of institutions each of which has an associated finite set of roles, and system participants might play one or more of these roles in one or many of the system institutions. Specifying an organisation can be done through specifying the inter-agent relationships that exist within this organisation [15]. In MSMAS, these inter-relationships specifications are centered around the abstract roles the system participants can play and how these roles relate to each other. Role specification allows for defining behaviour patterns in an abstract way, independent from each individual system participant. In this sense, roles are considered as system participant types, so when a system participant takes part in an institution and plays one of the institution roles, it should conform to that pattern of behaviour. All system agents that adopt the same role are normally granted the same rights and duties, and are expected to obey to the same restrictions applied to that role. Declarative specification allows the identification of an arbitrary range of relations, but in MSMAS we restrict ourselves to the following role types, as seen in the role/role relations in Figure 1.

1. **Sequential Roles (SR)**: these are the pairs of roles where the the system participant is required to play the first role before being allowed to play the second one. An example from Figure 1 is when an agent has to be *Catalogue Manager* before being *Stock Manager*.
2. **Joint Roles (JR)**: these are pairs of roles where the system participant is required to play both, but one after another in a specified order, or neither. The first role is considered a precondition to place the second role. An example is the requirement in a marketplace that a type of seller should fulfill their customers' orders themselves, meaning they have to play the role of Shipper after playing being Seller.
3. **Coupled Roles (CR)**: these are pairs of roles that are coupled together where the system participant is required to play both or none, but in either order: once one of them played the other one needs to be played. An example is the requirement in a marketplace that product meta data provider is the same as the inventory data provider. *CR* allows for concurrency where an agent is required to play multiple roles at the same time.
4. **Disjoint Roles (DR)**: these are mutually exclusive roles, where only one of them can be played by a system participant at any point of time, and once the system participant plays that role it can not play the other role. An example of this is when you have a Coder and Code reviewer, and the requirement that one can not review his own code (*four eyes principle*).
5. **Amicable Roles (AR)**: these are the pairs of roles where the system participant can play one or many of them at the same time without raising any conflict.

The set of all Institutional Roles IR is then represented as: $IR_{msmas} = \langle SR \cup JR \cup CR \cup DR \cup AR \cup HR \rangle$ and each role set in turn is defined as: $R_{inst} = \langle inst, R, R_{rel} \rangle$ where R is the set of roles that belong to institution $inst$ and R_{rel} is the set of relations between these roles in R .

Business Processes. In MSMAS there are two types of business processes models: (i) Composite Business Process (CBP)¹: which is a System Conceptual Plan (SCP) that describes which business processes and/or business activities are needed for the achievement of a Composite System Goal (CSG)²³, and (ii) Basic Business Process (BBP): which contains the detailed specifications of the actual business activities that lead to the achievement of a Basic System Goal (BSG)⁴. Each BSG goal can be achieved through the execution of one or more activities as defined through the system design. MSMAS allows designers to assign any DECLARE-style constraint/relation to any pair of business activities within any BBP⁵. Relations within the context of CBPs however are limited to only four types, as shown in Figure 1 BP/BP relations, where we identify the following types for CBPs as conceptual plans:

1. **Sequential Business Processes (SBP)**: these are the pairs of business processes/activities that must be executed consecutively.

¹ Composite Business Process was called in previous publications as Specific Business Process.

² Composite System Goal was called in previous publications as Specific System Goal.

³ Composite System Goal: is a functional goal achievable by one or more business process.

⁴ Basic System Goal: leaf of system goals tree, achievable by one or more business activities.

⁵ All the DECLARE relation formulae, notation and SCIFF mapping appears in [10].

2. **Joint Business Processes (JBP)**: these are pairs of business processes/activities where both are required to be executed, but one after another in a specified order.
3. **Coupled Business Processes (CBP)**: these are pairs of business processes/activities where both are required to be executed, but in no specific order.
4. **Disjoint Business Processes (DBP)**: these are mutually exclusive business processes/activities, where only one may be executed, and once this has occurred the other process/activity can not be executed.
5. **Amicable Business Processes (ABP)**: these are the pairs of business processes/activities that can be executed freely without raising any conflict.

The set of all possible Business Processes/Activities SBP is represented as:

$$SBP_{msmas} = \langle SCP_{msmas} \cup SEP_{msmas} \rangle$$

where the set of all MSMAS System Business Process *CBP* is the set of both System Conceptual Plans *SCP* and the set of System Executable Plans *SEP*:

$$SCP_{msmas} = \langle CBP, SBA, SCP_{rel}, CSG \rangle$$

$$SEP_{msmas} = \langle SBA, SEP_{rel}, BSG \rangle$$

where *SBA* is the set of business activities that lead to the achievement of the System Basic Goals *BSG* and *SEP_{rel}* is the set of relations between these system activities.

Communications Protocols. A communication protocol in MSMAS is a set of one or more messages sent from one system participant to another. We define three types of communication messages:

1. **Inform Message (IM)**: where a system participant sends information in some form such as a belief, a file, etc to another; the sender does not expect a response and the recipient does not expect to reply. This message type is useful for lightweight communications scenarios where acknowledging delivery is not required or essential.
2. **Offer Message (OM)**: where a system participant offers to send some information such as a belief, a file, etc to another system participant. The recipient is expected to respond by accepting or rejecting this offer; if accepted an inform message should follow.
3. **Request Message (RM)**: where a system participant asks for some information from another. A response is required accepting or rejecting the request: either way the response is an inform message or another request message.

In MSMAS a communication protocol *CommuProt* is defined as:

$$CommuProt_{msmas} = \langle Msg_{prot}, Msg_{rel}, IR_{prot} \rangle$$

where a communication protocol *CommuProt_{msmas}* is the set of communication messages *Msg_{prot}* that are exchanged between the system participants playing the institutional *IR_{prot}* roles according to the constraints set by the set of relations *Msg_{rel}* between the pairs of these messages and:

$$Msg_{msmas} = \langle IM \cup OM \cup RM \rangle$$

$$\text{where } msg_{msmas} = \langle sender, recipient, msgContent, timeStamp \rangle$$

3 The SCIFF Framework

SCIFF is a logic programming framework originally proposed by Alberti et al [1]. It is based on Abductive Logic Programming (ALP) [9]. An ALP is a triple $\langle P, A, IC \rangle$,

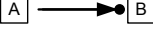
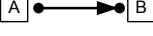
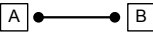
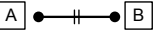

DECLARE Notation	DECLARE Visual notation	MSMAS Role/Role Relation	MSMAS BP/BP Relation
Precedence Relationship: If B is performed A should have been performed before it		Sequential Roles: The system participant has to play Role B only after playing Role A	Sequential BPs: BP B has to be executed only after the execution of BP A
Succession Relationship: every execution of A should be followed by the execution of B and each B should be preceded by A		Joint Roles: The system participant must play Role B after playing Role A , and must have played Role A in order to play Role B	Joint BPs: BP B must be executed after the execution of BP A , and BP A must have been executed to start executing BP B
Coexistence Relationship: If either A or B is performed, the other one has to be executed as well.		Coupled Roles: The system participant has to play both Role A and Role B	Coupled BPs: Both BP A and BP B have to be executed
Not Coexistence Relationship: If one of A or B is performed, the other one can not be executed.		Disjoint Roles: If the system participant plays Role A then Role B can not be played, and vice versa.	Disjoint BPs: If BP A has been executed then BP B can not be executed, and vice versa.
No constraint		Amicable Roles: The system participant can play any or both of Role A and Role B without any restriction	Amicable BPs: Any or both of BP A and BP B can be executed without any restriction as long as their specified pre conditions -if any- are met

Fig. 1. DECLARE Notation and its Mapping to MSMAS Role/Role relation concepts

where A is a set of predicates, named *abducibles*, P is a logic program that uses predicates in A but does not define them, and IC is a set of integrity constraints.

Reasoning in abductive logic programming is a goal-directed task (G , a goal), and amounts to finding an explanation set Δ of (ground) abducible predicates, such that: $P \cup \Delta \models G$ and $P \cup \Delta$ is consistent. The set IC of integrity constraints constrains the explanations Δ for the goal G , through the additional requirement $P \cup \Delta \models IC$.

SCIFF leverages on ALP to constrain the dynamics of an event-based system, such as the interaction between multiple agents [1] or the execution of a business process [12]. In particular, SCIFF instantiates the ALP triple $\langle P, A, IC \rangle$ as follows:

- A is constituted by special predicates denoting expectations about (un)desired events;
- P is a knowledge base used to capture the static knowledge of the targeted system;
- IC is used to relate the occurrence of events to expected events, thus defining which are the events that are expected to occur when a certain trace of BP events is observed.

Events. In SCIFF there is a clear distinction between the description of an event and the occurrence of said event. In fact, an event is represented as a term, whereas an event that has happened is an atom $\mathbf{H}(\text{Event}, \text{Time})$ where *Event* is a *Term* and *Time* is an integer denoting the time at which that event happened. Ground happened events are used to represent a (partial) execution trace of the system, enumerating the relevant events and their timestamps, whereas happened events with variables are used to denote a class of matching ground happened events. For example, $\mathbf{H}(\text{inform}(\text{john}, \text{mary}, \text{call_code}(123)), 5)$ denotes that *john* informed *mary* at time 5 that the call code has value 123. Whereas, $\mathbf{H}(\text{inform}(X, \text{mary}, \text{call_code}(C)), T)$ models that some agent X informed *mary* at a certain time T that the call code has value C .

As well as happened events, SCIFF supports the modelling of (un)desired courses of interaction by introducing the notion of *expected* events, making it possible to explicitly describe what is expected (not) to happen. Expectations can be either positive

$\mathbf{E}(\text{Event}, \text{Time})$ or negative $\mathbf{EN}(\text{Event}, \text{Time})$. The intuitive quantification for the variables possibly contained in the expectations is existential for positive expectations, and universal for negative expectations. For example, $\mathbf{E}(\text{inform}(X, \text{mary}, \text{call_code}(C)), T)$ models that it is expected that someone informs *mary* about the call code at some point in time, whereas $\mathbf{EN}(\text{inform}(X, \text{mary}, \text{call_code}(C)), T)$ means that no agent can ever inform *mary* about call codes. The full SCIFF event syntax appears in [1].

SCIFF Integrity Constraints. In the SCIFF framework, integrity constraints (*ICs*) are used to express behavioural rules interconnecting happened events with expectations, to represent the expected and forbidden courses of interaction when a given pattern of happened events is found in the current system trace. Technically, they are (forward) implications of the form $\beta(\mathbf{X}) \rightarrow \gamma(\mathbf{X}, \mathbf{Y})$, where $\beta(\mathbf{X})$ is a conjunction of literals, i.e., of (partially grounded) happened/expected events and other predicates, and $\gamma(\mathbf{X}, \mathbf{Y})$ is a disjunction or conjunction of expectations and other predicates. Intuitively, variables \mathbf{X} are universally quantified with the entire implication as scope, whereas variables \mathbf{Y} are existentially or universally quantified depending on whether they appear inside positive or negative expectations (for a full account of quantification, see [1]). Predicates are used to constrain further the matching events, and include Constraint logic programming (CLP)⁶ constraints. When applied to time variables, CLP constraints are particularly useful for imposing metric temporal conditions on happened/expected events. For example, the integrity constraint:

$$\mathbf{H}(\text{create_call}(X, C), T) \wedge \text{friend}(X, Y) \rightarrow \mathbf{E}(\text{inform}(X, Y, \text{call_code}(C)), T_2) \wedge T_2 < T + 10$$

states that whenever agent *X* creates a call with code *C*, *X* is expected to inform each of her friends about the value of the call code within 10 time units. Once again, for the full SCIFF social integrity constraint syntax, see [1].

SCIFF Knowledge Base. SCIFF *ICs* can capture the dynamic aspects of a system by interconnecting the observed and expected courses of interaction. However, they are not meant to represent the static knowledge that might be needed to describe the system independently of its dynamics. The SCIFF framework allows the definition of this type of knowledge inside a knowledge base (*KB*). The *KB* can be used to list facts known about the domain under study (such as the extension of the *friend* predicate used in the aforementioned sample integrity constraint), or to encode complex derivation rules modeled as logic programming clauses. Such derivation rules could also employ happened and expected events to provide a-priori definitions for knowledge related to the system dynamics. The full syntax for SCIFF knowledge base terms is given in [1].

Compliance in SCIFF. We now describe the declarative semantics of SCIFF, which builds upon the semantics of ALP and extends it so as to capture the meaning of expectations. In particular, SCIFF declaratively captures the notion of *compliance* of a system execution trace with the modelled specification. This is done by considering positive and negative expectations as abducible predicates, and by introducing the notion of *fulfillment*. Starting from the knowledge base and the set of happened events

⁶ A Constraint Logic Program is a logic program that contains constraints in the body of clauses.

contained in the analyzed trace of the system (which extends the knowledge base with information about the dynamics), expectations are hypothesized consistently with the *ICs* and with an expectation-consistency rule stating that no event can be expected to happen and not to happen at the same time. A positive (respectively negative) expectation is then judged as fulfilled (respectively violated) if there exists a corresponding matching happened event in the trace. This can be considered as a sort of *hypothesis confirmation* step, where the hypothesized courses of execution match with an actual behaviour.

This declarative notion of compliance has an operational counterpart in the SCIFF proof procedure, which concretely realizes an inference mechanism to 1. dynamically acquire happened events reporting about the evolution of the system dynamics, 2. use the modeled knowledge base and integrity constraints so as to generate expectations about the courses of execution, and 3. match expectations with happened events, deciding their fulfillment. Execution traces which fulfill all the generated expectations are then deemed as *compliant* with the specification.

An extension of the SCIFF proof procedure, called *g-SCIFF*, can be used to prove properties of the model at design time, i.e., without having an explicit trace of the system [12]. Given a property, *g-SCIFF* tries to generate a (partially specified) trace showing that the property can be satisfied while respecting all the modeled *ICs*. Intuitively, this is done by transforming every pending positive expectation into a corresponding happened event, checking that no negative expectation becomes violated.

Finally, we observe that while termination of the proof procedures cannot be guaranteed in general, all the techniques developed to check termination of (abductive) logic programs can be seamlessly applied to SCIFF (see, e.g., [12,10] for a discussion on termination conditions when reasoning on extended DECLARE).

4 MSMAS Semantics in SCIFF

In this section we establish a correspondence between MSMAS and SCIFF, consequently enabling the exploitation of the reasoning capabilities of the SCIFF framework to systems modelled with MSMAS. The translation is inspired by [11,10].

4.1 Events in MSMAS

In MSMAS, events reflect the execution of business processes, as well as the dynamics of institutions and of agent interaction. In particular, a system execution is understood by MSMAS in terms of the following events.

Institutional events are triggered when a system participant plays a defined institutional role. For example:

*playRole(*Agent1, (*marketplaceInstitution*, *seller*))

where the agent (*Agent1*) plays the role (*seller*) within the defined institution (*marketplaceInstitution*).

Communications events are triggered when a system participant sends a message within a defined communication protocol. For example:

requestMsg((*buyProductProtocol*, *customer1*, *seller5*,
getprice(*ean* : 9782735638938)))

where the (*customer1*) agent sends request message which is part of (*buyProductProtocol*) communication protocol to get the price of product with ID/EAN (9784431540816) to get the price from (*seller5*) agent.

Business process events are triggered when a system participant starts/ends the execution of an activity within a defined basic business process. For example:

$$\text{startActivity}(\text{checkForNewSSL}(\text{updateFileBP}, (\text{updateAvailabilityCP})), \\ \text{Agent1}, (\text{updateFileBSG}, \text{false}), (\text{timeForUpdate}, \text{true})))$$

where the (*Agent1*) agent starts the execution of (*checkForNewSSL*) activity within the defined basic process (*updateFileBP*) which in turn is a step in the conceptual plan (*updateAvailabilityCP*) to achieve the basic system goal (*updateFileBSG*) with inputs/preconditions (*TimeForUpdate*) with value (*true*). or

$$\text{endActivity}(\text{checkForNewSSL}(\text{updateFileBP}, (\text{updateAvailabilityCP})), \\ \text{Agent1}, (\text{updateFileBSG}, \text{false}), (\text{fileFound}, \text{true})))$$

where the agent *Agent1* ends the execution of activity *checkForNewSSL* within the defined basic process *updateFileBP*, which in turn is a step in the conceptual plan set CP, that contains conceptual plan *updateAvailabilityCP*, to achieve basic system goal *updateFileBSG* with outputs/postconditions *fileFound* with value *true*.

4.2 Institutional Role/Role Relation Formalisation

Given a MSMAS model, each Role/Role relation present in the model is captured as a corresponding fact of the type: *role_role_relation(A, B, Type)* For example, *role_role_relation(Registered_User, Seller, sequential_roles)*, expresses that a precedence/sequential role relation holds between *Registered_User* and *Seller* roles. All these facts are grouped together inside an “institution” knowledge base \mathcal{KB}_{inst} .

The Role/Role relations described in Table 1 are then formalized by means of *ICs* that follow the DECLARE to SCIFF translation presented in [11,10]. Such constraints are grouped together into an integrity constraint set \mathcal{IC}_{inst} .

Sequential Roles. A sequential role relation is represented by the following *IC*:

$$\text{role_role_relation}(A, B, \text{sequential_roles}) \\ \wedge \mathbf{H}(\text{play_role}(\text{SystemParticipant}, (\text{Institution}, B), T_B)) \\ \rightarrow \mathbf{E}(\text{play_role}(\text{SystemParticipant}, (\text{Institution}, A), T_A)) \wedge T_A < T_B.$$

Notice that the constraint is instantiated for every Role/Role relation of type *sequential_roles* contained into \mathcal{KB}_{inst} .

Joint Roles can be formalised using the following *ICs*:

$$\text{role_role_relation}(A, B, \text{joint_roles}) \\ \wedge \mathbf{H}(\text{play_role}(\text{SystemParticipant}, (\text{Institution}, A), T_A)) \\ \rightarrow \mathbf{E}(\text{play_role}(\text{SystemParticipant}, (\text{Institution}, B), T_B)) \wedge T_B > T_A. \\ \text{role_role_relation}(A, B, \text{joint_roles}) \\ \wedge \mathbf{H}(\text{play_role}(\text{SystemParticipant}, (\text{Institution}, B), T_B)) \\ \rightarrow \mathbf{E}(\text{play_role}(\text{SystemParticipant}, (\text{Institution}, A), T_A)) \wedge T_A < T_B.$$

Coupled Roles can be captured as joint roles, but without imposing any ordering constraint on the event timestamps:

$$\begin{aligned}
 & \text{role_role_relation}(A, B, \text{coupled_roles}) \\
 & \wedge \mathbf{H}(\text{play_role}(\text{SystemParticipant}, (\text{Institution}, A), T_A)) \\
 & \quad \rightarrow \mathbf{E}(\text{play_role}(\text{SystemParticipant}, (\text{Institution}, B), T_B)). \\
 & \text{role_role_relation}(A, B, \text{coupled_roles}) \\
 & \wedge \mathbf{H}(\text{play_role}(\text{SystemParticipant}, (\text{Institution}, B), T_B)) \\
 & \quad \rightarrow \mathbf{E}(\text{play_role}(\text{SystemParticipant}, (\text{Institution}, A), T_A)).
 \end{aligned}$$

Disjoint Roles are formalized by means of negative expectations:

$$\begin{aligned}
 & \text{role_role_relation}(A, B, \text{disjoint_roles}) \\
 & \wedge \mathbf{H}(\text{play_role}(\text{SystemParticipant}, (\text{Institution}, A), T_A)) \\
 & \quad \rightarrow \mathbf{EN}(\text{play_role}(\text{SystemParticipant}, (\text{Institution}, B), T_B)).
 \end{aligned}$$

4.3 Business Processes and Business Activities Relation Formalisation

In MSMAS, the modelling of Composite Business Processes (CBPs) means representing the conceptual plans broken down into the necessary steps to achieve one specific system goal. Meanwhile the actual executable plans are those that are modelled as Basic Business Processes (BBPs), where the plan steps are business activities (BAs). We allow the system designer to set constraints between BP/BP, BP/BA, and BA/BA at any level, whether part of a conceptual plan or of an executable plan: the only difference is that a BP/BP or BP/BA relationship at the conceptual plan level is inherited all the way down through all sub-processes/sub-plans. BP/BP or BP/BA relations are restricted to the types described in §2 (Sequential Business Processes, Joint Business Processes, Coupled Business Processes, and Disjoint Business Processes) and are formalised in the KB_{SCP} as a fact of the type: $scp_scp_relation(A, B, Type)$. For example, $scp_scp_relation(\text{checkForNewSSL}, \text{publishSupplierUpdate}, \text{sequential_business_process})$, expresses a $sequential_business_process$ relation between the checkForNewSSL and $\text{publishSupplierUpdate}$ business processes.

The formalisation of the relations as integrity constraints follow the same method as for role/role relations in section 4.2. An interesting pattern is the one of *sequential processes*, which can be represented by means of a DECLARE *chain response*, in turn captured in SCIFF as follows:

$$\begin{aligned}
 & scp_scp_relation(A, B, \text{sequential_business_process}) \\
 & \wedge \mathbf{H}(\text{execute}(BP_A, \text{SystemParticipant}, \text{Beliefs}), T_A) \\
 & \quad \rightarrow \mathbf{E}(\text{execute}(BP_B, \text{SystemParticipant}, \text{Beliefs}), T_B) \wedge T_B > T_A \\
 & \quad \wedge \mathbf{EN}(\text{execute}(_, _, _), T_x) \wedge T_x > T_A \wedge T_x < T_B.
 \end{aligned}$$

The full set of DECLARE constraints are supported at the BA/BA level, where these business activities are steps within an executable plan. For a comprehensive treatment of such constraints in SCIFF, see [10].

4.4 Communication Protocols Relation Formalisation

Communication protocols serve as a vehicle for enabling the development of interoperable agents and they mainly facilitate negotiation, cooperations and coordination

among all different system participants according to the system design. In MSMAS as explained in Section 2, using the identified three types of messages the system designer can create any custom communication protocol of any length of a finite set of messages. Only Offer Message and Request Message types do require a response when sent, so effectively a request message has a succession relationship with a response message i.e. it is a **Joint Relationship** between two messages that can be formalised in the KB_{MSG} as a fact of the type: $msg_msg_relation(protocol, A, B, Type)$. For example, $msg_msg_relation(getPrices, submitFileUpdate, submitSecurityToken, joint_message)$, expresses a *joint_message* relation between *submitFileUpdate* and *submitSecurityToken* messages in the communication protocol *getPrices*.

The formalisation of the relations as integrity constraints follow the same method as per previous examples in previous sections. **Joint Communication Messages** can be formalised using the following integrity constraints:

$$\begin{aligned}
& msg_msg_relation(prot_1, requestMsg, informMsg, joint_message) \\
& \wedge \mathbf{H}(requestMsg(prot_1, SystemParticipant_A, SystemParticipant_B, Content), T_A) \\
& \rightarrow \mathbf{E}(informMsg(prot_1, SystemParticipant_B, SystemParticipant_A, Content), T_B) \\
& \wedge T_B > T_A. \\
& msg_msg_relation(prot_1, requestMsg, informMsg, joint_message) \\
& \wedge \mathbf{H}(informMsg(prot_1, SystemParticipant_B, SystemParticipant_A, Content), T_B) \\
& \rightarrow \mathbf{E}(requestMsg(prot_1, SystemParticipant_A, SystemParticipant_B, Content), T_A) \\
& \wedge T_A < T_B.
\end{aligned}$$

4.5 Reasoning about MSMAS Models

By putting together the translation principles presented above, we obtain a full SCIFF specification constructed as follows:

$$\begin{aligned}
& \mathcal{P}_{msmas} \equiv \langle KB_{msmas}, \{\mathbf{E}/2, \mathbf{EN}/2\}, \mathcal{IC}_{msmas} \rangle \\
\text{where } & KB_{msmas} \triangleq KB_{inst} \cup KB_{prot} \cup KB_{act} \\
\text{and } & \mathcal{IC}_{msmas} \triangleq \mathcal{IC}_{inst} \cup \mathcal{IC}_{prot} \cup \mathcal{IC}_{act}
\end{aligned}$$

The SCIFF and g-SCIFF proof procedure can consequently be applied to reason about MSMAS models. Notably, the joint application of these proof procedures covers the entire lifecycle of a system: at design time, the SCIFF proof procedure can be used to check compliance of simulated event traces, while g-SCIFF can be applied to verify whether the MSMAS model of the system meets some desired properties; at runtime, the SCIFF proof procedure can be employed to monitor the running system and check whether it fulfils the generated expectations; a-posteriori, the same approach can be employed to analyze complete traces representing past system executions.

More specifically, the correspondence between MSMAS and SCIFF gives an expectation-based declarative semantic for MSMAS, providing a formal notion of *compliance* between an execution trace of the system (also called *instance* of the specification) and the constraints obtained from the MSMAS model. Intuitively, given a set **HAP** of happened events, \mathcal{P}_{msmas} leads to formulate an abductive set **EXP** that contain positive and negative expectations, which reflect the events that are expected (not) to occur in the state of affairs obtained after the execution of the events in **HAP**. In this

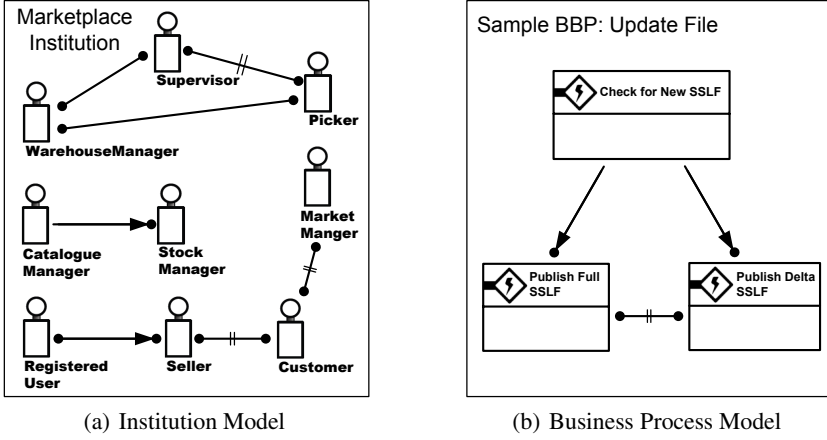


Fig. 2. MSMAS example models

respect, we take advantage of the declarative semantics of *SCIFF* to tackle three basic reasoning tasks: *consistency*, *fulfillment*, and *conformance*.

Consistency states that a MSMAS event cannot be expected to happen and expected not to happen at the same time. Technically, for each (ground) MSMAS event e and timestamp t , consistency requires that $\{\mathbf{E}(e, t), \mathbf{EN}(e, t)\} \not\subseteq \mathbf{EXP}$. Notice that consistency is not checked by the proof procedures by effectively grounding the expectations, but by using variables and CLP constraints to maintain an intensional, “symbolic” representation of (classes of) expectations, using constraint-solving to detect clashes between positive and negative expectations.

Fulfillment expresses the semantics of expectations. In particular, we say that a positive expectation $\mathbf{E}(e, t) \in \mathbf{EXP}$ is fulfilled by a set \mathbf{HAP} of happened events if and only if $\mathbf{H}(e, t) \in \mathbf{HAP}$, i.e., a corresponding happened event has occurred. Specifically, a negative expectation $\mathbf{EN}(e, t) \in \mathbf{EXP}$ is fulfilled by a set \mathbf{HAP} of happened events if $\mathbf{H}(e, t) \notin \mathbf{HAP}$, i.e., no corresponding happened event has occurred. Furthermore, we say that \mathbf{EXP} is fulfilled by \mathbf{HAP} if every expectation in \mathbf{EXP} is fulfilled by \mathbf{HAP} .

Conformance combines the notion of consistency and fulfillment to characterize whether a trace of the system respects all the constraints imposed by the MSMAS model. In particular, given a goal G and a complete trace of the system \mathbf{HAP} , we say that \mathbf{HAP} *conforms to* the MSMAS model satisfying G if:

$$\begin{aligned}
 \mathcal{KB}_{msmas} \cup \mathbf{HAP} \cup \mathbf{EXP} &\models G \\
 \mathcal{KB}_{msmas} \cup \mathbf{HAP} \cup \mathbf{EXP} &\models \mathcal{IC}_{msmas} \\
 &\mathbf{EXP} \text{ is consistent} \\
 &\mathbf{EXP} \text{ is fulfilled by } \mathbf{HAP}
 \end{aligned}$$

Looking closely at the sample institution model in Figure 2, we observe that the specifications of *WarehouseManager*, *Supervisor* and *Picker* eventually lead to an inconsistency as soon as an agent start to play one of these roles: *WarehouseManager* and *Supervisor* are joint roles, and so are *WarehouseManager* and *Picker*, which implies that also *Supervisor* and *Picker* have to be joint roles, while the model constrains them

to be disjoint. This inconsistency can be detected by the *SCIFF* proof procedure when a partial trace of the system is analyzed. With *g-SCIFF*, instead, the problem can, e.g., be detected when the modeller asks the following query: *is it possible for an agent to play the role of Picker?* Observe that a negative answer to this query would seriously question the correctness of the *MSMAS* model, because it would attest that *Picker* is always an “empty” role in any possible execution. In *g-SCIFF*, such a query can be expressed in terms of the goal: $\mathbf{E}(\text{play_role}(SP, (I, picker), T))$. To answer this query, *g-SCIFF* tries to generate a (partially specified) trace that conforms to the *MSMAS* model and at the same time satisfies the goal. In particular, starting from the expectation mentioned in the goal, the proof procedure generates a happened event that fulfils the expectation: $\mathbf{H}(\text{play_role}(SP, (I, picker), T))$. This event states that at some time T , an agent SP will indeed play the *Picker* role in the context of some institution I . Due to the two constraints attached to the *Picker* role in Figure 2(a), this in turn triggers the generation of two expectations: one stating that the *WarehouseManager* role must also be played by that agent ($\mathbf{E}(\text{play_role}(SP, (I, warehouse_manager), T_2))$), and the other stating that the *Supervisor* role can never be played by that agent ($\mathbf{EN}(\text{play_role}(SP, (I, supervisor), T_3))$, where T_3 is universally quantified). To fulfill the first expectation, *g-SCIFF* generates a corresponding happened event, which however triggers a further positive expectation about the fact that the agent must also play, sooner or later, the *Supervisor* role ($\mathbf{E}(\text{play_role}(SP, (I, supervisor), T_4))$, where T_4 is existentially quantified). This expectation clashes with the negative expectation generated before, leading to inconsistency and, in turn, to a negative answer for the posed query.

5 Discussion and Future Work

As far as this presentation is concerned, we do make some simplifying assumptions about scenarios being modelled of which we highlight: (i) all system goals are compatible in that they lead to the achievement of the general goal of the system. The designer can add conflicting goals if the intention is to construct a competitive, however *MSMAS* and its tool does not currently support the identification of conflicting system goals and does not provide any guidance as yet, on how to resolve such issues. (ii) it is the designer’s responsibility to specify how to manage the registration and deregistration of system participants with respect to an institution. The registration process is normally used as a mechanism for declaring the role(s) that a system participant intends to play and by so doing, the institution governor may validate the participant’s eligibility, prior to starting any activity with that role.

Meta-models for norm-aware MAS are an active research area as an incomplete selection from the literature demonstrates. The *Alive* project [2] used model-driven development to deploy agents and services in the context of an organizational model defined in *Opera*, which expresses norms in terms of states to be maintained or avoided. However, *Opera* lacks an activity model, making run-time validation difficult. The very detailed formalisation of *EIDE* [3] takes what is arguably a more low-level approach to capturing the quite complex semantics, particularly those associated with scenes and transitions, using the *Z* specification language. While this provides a high degree of precision, there is no associated meta-model, so although there are tools to generate code automatically, they do not have that formal backing. A further issue is the relatively stylised nature of *EIDE* agents, whose actions have historically been highly

constrained through the use of governors that ensure non-compliant actions never occur. Ghorbani [8] provides a different perspective, in which a meta-model is defined and used for model-driven development in the context of agent-based simulations, informed by Ostrom's IAD framework. While this emphasizes the role of norm and institution in the governance of agent behaviour, the meta-model reflects the research objectives of modelling social structures and their evolution. Against this background – and much literature for which there is not room here – we suggest that (i) the novel combination of a business-oriented institutional meta-model, designed to support self-management in use, (ii) with a design- and run-time formal proof mechanism, (iii) provides a new perspective on formal software engineering of MAS and contributes towards the evolution of and the debate on the application of meta-models and model-driven development in MAS. Future plans for MSMAS include the enhancement of its metamodel to be Model Driven Development (MDD) compliant and to investigate and include the modelling of multiple organisations/institution interactions.

References

1. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Verifiable agent interaction in abductive logic programming: The sciff framework. *ACM Trans. Comput. Logic* 9(4), 29:1–29:43 (2008)
2. Aldewereld, H., Padget, J., Vasconcelos, W., Vázquez-Salceda, J., Sergeant, P., Staikopoulos, A.: Adaptable, Organization-Aware, Service-Oriented Computing. *IEEE Intelligent Systems* 25(4), 26–35 (2010)
3. d'Inverno, M., Luck, M., Noriega, P., Rodríguez-Aguilar, J.A., Sierra, C.: Communicating open systems. *Artificial Intelligence* 186, 38–64 (2012)
4. Baldoni, M., Baroglio, C., Mascardi, V., Omicini, A., Torroni, P.: Agents, Multi-Agent Systems and Declarative Programming: What, When, Where, Why, Who, How? In: Dovier, A., Pontelli, E. (eds.) *GULP. LNCS*, vol. 6125, pp. 204–230. Springer, Heidelberg (2010)
5. Elakehal, E.E., Padget, J.: Msmas: Modelling self-managing multi agent systems. *SCPE: Scalable Computing: Practice and Experience* 13(2), 121–137 (2012)
6. Elakehal, E.E., Padget, J.: A practical method for developing multi agent systems: APMD-MAS. In: Brazier, F.M.T., Nieuwenhuis, K., Pavlin, G., Warnier, M., Badica, C. (eds.) *Intelligent Distributed Computing V. SCI*, vol. 382, pp. 11–20. Springer, Heidelberg (2011)
7. Alberola, J.M., Such, J.M., Botti, V., Espinosa, A., Garcia-Fornes, A.: A scalable multiagent platform for large systems. *Computer Science and Information Systems* 10(1), 51–77 (2013)
8. Ghorbani, A., Bots, P., Dignum, V., Dijkema, G.: Maia: a framework for developing agent-based social simulations. *Journal of Artificial Societies and Social Simulation* 16(2), 9 (2013)
9. Kakas, A.C., Kowalski, R.A., Toni, F.: *Abductive logic programming* (1993)
10. Montali, M. (ed.): *Specification and Verification of Declarative Open Interaction Models. LNBIP*, vol. 56. Springer, Heidelberg (2010)
11. Montali, M., Pesic, M., van der Aalst, W.M.P., Chesani, F., Mello, P., Storari, S.: Declarative specification and verification of service choreographiess. *ACM Trans. Web* 4(1), 3:1–3:62 (2010)
12. Montali, M., Torroni, P., Chesani, F., Mello, P., Alberti, M., Lamma, E.: Abductive logic programming as an effective technology for the static verification of declarative business processes. *Fundamenta Informaticae* 102(3-4), 325–361 (2010)

13. Pesic, M., van der Aalst, W.M.P.: A declarative approach for flexible business processes management. In: Eder, J., Dustdar, S. (eds.) *BPM Workshops 2006*. LNCS, vol. 4103, pp. 169–180. Springer, Heidelberg (2006)
14. van der Aalst, W., Pesic, M.: Decserflow: Towards a truly declarative service flow language. In: Leymann, F., et al. (eds.) *The Role of Business Processes in Service Oriented Architectures*, Dagstuhl, Germany, vol. 06291, Dagstuhl Seminar Proceedings (2006)
15. Zambonelli, F., Jennings, N.R., Wooldridge, M.: *Organisational rules as an abstraction for the analysis and design of multi-agent systems* (2001)

Decentralised Cooperative Agent-Based Clustering in Intelligent Traffic Clouds

Jelena Fiosina, Maksims Fiosins, and Jörg P. Müller*

Clausthal University of Technology,
Institute of Informatics,

Julius-Albert Str. 4, D-38678, Clausthal-Zellerfeld, Germany

{Jelena.Fiosina,Maksims.Fiosins}@gmail.com, Joerg.Mueller@tu-clausthal.de

Abstract. Contemporary traffic management systems will become more intelligent with advent of future Internet technologies. The systems are expected to become more simple, effective and comfortable for users, but this transformation will require the development of both new system architectures as well as enhanced processing and mining algorithms for large volumes of cloud data. In this study, we consider a conceptual architecture of a cloud-based traffic management system that applied to a multi-modal journey planning scenario. For this purpose, it is necessary to process large amounts of travel-time information. Information is collected by cloud service providers and processed for future route planning. In this paper, we focus on the data clustering step in the data mining process. The data collection and processing require an appropriate clustering algorithm to aggregate similar data. In particular, we support a process where a particular service provider can request additional information from others to be used in the clustering function, requiring a decentralised clustering algorithm. We present a cloud-based architecture for this scenario, develop a decentralised cooperative kernel-density based clustering algorithm, and evaluate the efficiency of the proposed approach using real-world traffic data from Hanover, Germany.

Keywords: Cloud computing architecture, decentralised data processing and mining, multi-agent systems, kernel density estimation, clustering.

1 Introduction

Congested roads need to develop a new generation of Traffic Management Systems (TMS). These systems are very important for individual users, for example as drivers and pedestrians, business logistic operators and city public transport organizers. Such complex distributed systems can be well represented by multi-agent systems (MAS), which are based on multiple interacting and cooperating agents with intelligent behaviour.

* The research leading to these results has received funding from the EU 7th Framework Programme (FP7/2007-2013) under grant agreement No. PIEF-GA-2010-274881.

Future Internet capabilities such as cloud computing (CC) also influence TMS. CC aims at providing elastic services, high performance and scalable data storage to a large and ever increasing number of users [1]. CC systems provide large-scale infrastructure for high-performance computing and are dynamically adapted to user and application needs. Several common problems can be identified and several benefits obtained by the synergy between MAS and CC in an agent-based cloud computing (ABCC) paradigm. CC is mainly focused on the efficient use of computing infrastructure through reduced cost, service delivery, data storage, scalable virtualization techniques, and energy efficiency. In contrast, MAS are focused on the intelligent aspects of agent behaviour and their use in developing complex applications. In particular, CC can offer a very powerful, reliable, predictable and scalable computing infrastructure for the execution of MASs by implementing complex, agent-based applications for modelling and simulation. On the other hand, software agents can be used as basic components for implementing intelligence in clouds, making them more adaptive, flexible, and autonomic in resource management, service provisioning and large-scale application executions [14].

One of the key aspects of agent intelligence in ABCC is the capability to process and mine huge volumes of distributed data from various sources. Research activities have an inherent need to develop effective distributed data processing and mining algorithms for ABCC that take the needs and requirements of concrete traffic scenarios into account.

In this paper we consider travel time data processing that is distributed among cloud service providers. We focus on the problem of data clustering. Clustering is a descriptive data mining task used to partition a data-set into groups such that data objects in one group are similar to each other and are as different as possible from those in other groups. In cloud systems, data-sets are distributed among several providers, creating a need to develop distributed algorithms for data clustering. We develop a semi-parametric algorithm for distributed data clustering and consider some applications to ABCC-based intelligent TMS. We implement a kernel density (KD) technique based on contemporary computational statistics. It is a popular technique that reduces the search for clusters into a search for dense regions and allows finding arbitrary form clusters. This is accomplished using a so-called probability density function from which the given data set is assumed to have been generated [10].

The contribution of this study is the following: 1) a description of data processing stages in a cloud-based TMS that is applicable to a multi-modal journey planning scenario; 2) development of a novel decentralised cooperative agent-based KD clustering algorithm; 3) application of the proposed algorithm for the described scenario; 4) an experimental validation of the proposed clustering algorithm using real-world traffic data.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 formulates the considered TMS scenarios and presents the corresponding ABCC-based architecture. In Section 4, we present a KD clustering method. Section 5 suggests a decentralised, cooperative, agent-based KD

clustering algorithm and corresponding efficiency metrics. Section 6 reports the experimental results. The last section presents the conclusion and discusses the future work opportunities.

2 Related Work and State of the Art

By adapting to user and application needs, CC technologies, such as Future Internet and the Internet of Things (IoT), can enhance modern TMS and provide large-scale infrastructures for high-performance computing that are 'elastic' in nature [14]. CC technology is a worldwide priority research direction, and some authors (e.g., [12],[6]) have proposed work on next-generation TMS. Research areas are motivated and co-funded by private companies and municipalities in the areas of transport, logistics, communication and traffic management. Research in this area is still largely at the stage of scenario formulation and coordination protocols. One of the first cloud-based traffic network architecture, employing IoT components, has been proposed in [12].

Complex distributed systems are often modelled by MAS [7], [2]. Coupling CC with software agents opens new avenues for implementing intelligent cloud services. The convergence of interests between MAS, which require reliable distributed infrastructures, and CC systems, which needs intelligent software with dynamic, flexible, and autonomous behaviour, may result in new systems and applications [14]. Agents in such systems have the potential to be more intelligent and to possess special modules for intelligent data processing [10] and decision-making [7].

However, implementing a traffic cloud is far from easy. From an end user's point of view, the complexity of data and algorithms is hidden in the cloud. Users, ranging from traffic authorities to car drivers and automated components, expect to work with relatively simple web applications via mobile or embedded devices. These devices are permanently connected and can (theoretically) use all the information available from other users and system elements. A huge amount of available information must be found, collected, aggregated, processed and analysed for optimal decision-making and behaviour strategies. Such information is virtually centralized in cloud repositories, but should be managed physically in a decentralised fashion [6].

One of the key milestones in a path to more intelligent and up-to-date TMS is the continuous modification of existing methods and implementation of new modern technologies derived from the communication and data analysis fields. A decentralised regression forecasting model was considered in [5] and decision making methods in [7]. In this study, we focus on decentralised data clustering and the corresponding classification, which can be considered as a prerequisite step to implementation of forecasting and decision-making models.

In traffic research, the most popular clustering and classification problems are traffic state clustering [15] and participant behaviour clustering for group formation [11]. In this paper, we concentrate on clustering travel-time information,

which is part of the traffic state, trying to discover homogeneous traffic patterns that can be used with the common forecasting model.

KD clustering is a promising computational statistics tool that allows arbitrary shaped clusters to be discovered. Such non-parametric methods are well suited for exploring clusters, because no generative model of data is assumed. Instead, the probability density in the data space is directly estimated from data instances. Fast clustering, which is based on KD, was described by Hinnenburg and Gabriel [9]. The distributed (with a central authority) version of KD-based clustering (KDEC scheme) was considered in [10]. Another decentralised graph-oriented not KD clustering approach was presented in [13].

3 Cloud-Based TMS Architecture

We consider a future cloud-based TMS. The general architecture was proposed in [12] and then extended in [6]. The cloud-based TMS supposes the permanent connection of all participants and dynamically provides necessary services. The participants' applications that are run in the cloud are data-intensive, which necessitates the processing of large volumes of information to satisfy the participants' requests. In [6] we motivate the following data processing stages in a typical cloud-based TMS.

Stage 1: *Mining data from the IoT and its pre-processing.* All the participants of the cloud-based system have virtual representations as active IoT components (agents). The cloud system locates and collects the necessary data from different agents, and provides usual data mining operations (changes and outliers are detected, preliminary aggregation and dimensionality reduction are performed). The collected data are stored as historical information in the cloud and are used later as input data for ad-hoc network models (Stage 2).

Stage 2: *Ad-hoc network models.* The application-specific networks of virtual traffic participants are created, and the corresponding data models are used in order to estimate the important characteristics and parameters of these networks using the information collected in Stage 1 and for strategy optimization at Stage 3.

Stage 3: *Static decisions and initial strategy optimization.* Cloud applications use pre-calculated results of the ad-hoc network models from Stage 2 and the available historical information (including private information) about the traffic network to perform their pre-planning tasks. These models are also checked in the digital traffic network.

Stage 4: *Dynamic decisions and strategy update.* The pre-planned tasks from Stage 3 are executed, and updates are made according to the dynamic real-time situation extracted from the virtual agents. The aggregation of the pre-planned data and strategies with the dynamic ones is the most important problem at this stage.

We consider a dynamic multi-modal journey planning scenario [6]. In this scenario, the TMS helps travellers to plan and adjust a multi-modal, door-to-door journey in real-time. To provide recommendations to travellers, the

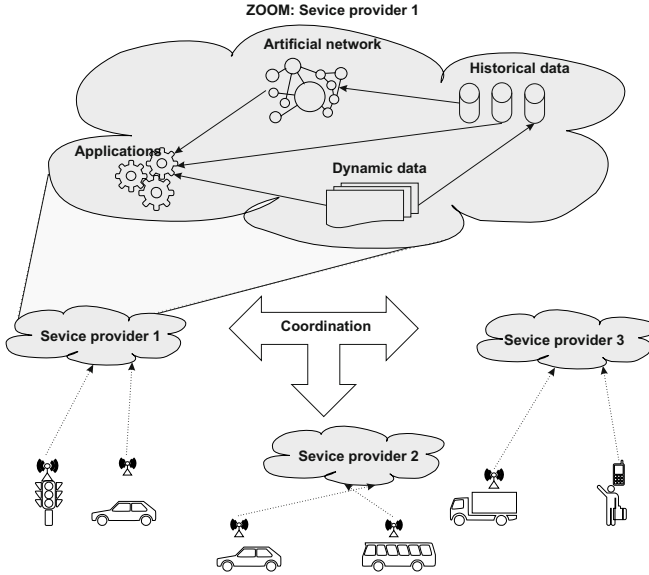


Fig. 1. System architecture

cloud-based TMS collects travel-time data in Stage 1 and organizes a historical travel information repository. In Stage 2 the travel times are continuously estimated for the travel maps. Stage 3 supposes construction of pre-defined routes for popular origin-destination pairs on the maps. In Stage 4 the actual information is taken into account and routes are updated correspondingly.

Note that there is no single 'central' cloud for this purpose. Instead, many *providers* may offer similar multi-modal journey planning services. They each collect information from subscribed travellers, and then create their own independent repositories, ad-hoc networks, and travel recommendations.

The providers are motivated by self-interest; their main goal is to maximize profit. To achieve this goal, they must balance two conflicting aspects of data processing. On the one hand, the providers are interested in maintaining a unique repository that provides clients with better services than those offered by competitors. On the other hand, the service providers are also interested in selling traveller information for profit (both to clients and to other providers).

After the creation of a (physical or virtual) data repository at Stage 1, each provider should process data in Stage 2 to prepare information for decisions in Stage 3. An important problem at Stage 2 is that of travel-time estimation for travel segments in the network. Different models, for example regression models, can be used for estimation [5]. However, to produce quality regression results, a single regression model should be used for each cluster, with potentially different models used across different clusters. Clustering of collected data is therefore one of the first tasks performed in Stage 2. In the next Section we describe a KD

technique of contemporary computational statistics, that allows finding clusters of arbitrary form [10], [9].

4 Kernel Density (KD) Clustering

Now let us formulate the clustering problem and describe the KD clustering algorithm. Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, $\mathbf{x}_i \in R^d$ be a dataset to be clustered into k non-overlapping subsets S_1, S_2, \dots, S_k .

Non-parametric clustering methods are well suited for exploring clusters of arbitrary form without building a generative model of the data. KD clustering consists of a two-step procedure: estimation and optimisation. During the estimation step, the probability density of the data space is directly estimated from data instances. During the optimisation step, a search is performed for densely populated regions in the estimated probability density function.

Let us formalize the estimation step. The density function is estimated by defining the density at any data object as being proportional to a weighted sum of all objects in the data-set, where the weights are defined by an appropriately chosen kernel function [10].

A KD estimator is

$$\hat{\psi}^{[\mathbf{X}]}(\mathbf{x}) = \frac{1}{N} \sum_{\mathbf{x}_i \in \mathbf{X}} |\mathbf{H}|^{-1} K(\mathbf{H}^{-1} \|\mathbf{x} - \mathbf{x}_i\|) = \frac{1}{N} \sum_{\mathbf{x}_i \in \mathbf{X}} K_{\mathbf{H}}(\|\mathbf{x} - \mathbf{x}_i\|), \quad (1)$$

where $\|\mathbf{x} - \mathbf{x}_i\|$ is a vector of coordinate distances between \mathbf{x}_i and \mathbf{x} , \mathbf{H} is a *bandwidth* matrix, $K(\mathbf{x})$ is a kernel function, $K_{\mathbf{H}}(\bullet) = |\mathbf{H}|^{-1} K(\mathbf{H}^{-1}\bullet)$ [8].

$K(\mathbf{x})$ is a real-valued, non-negative function on R^d and has finite integral over R^d . We use the multivariate Gaussian function in our study: $K(\mathbf{x}) = (2\pi)^{-d/2} \exp(-\frac{1}{2}\mathbf{x}^T \mathbf{x})$. The bandwidth matrix \mathbf{H} is a $d \times d$ positive-definite matrix that controls the influence of data objects and smoothness of the estimate. If no information is available with regard to correlation between factors, a diagonal matrix $\mathbf{H} = \text{diag}(h_1, \dots, h_d)$ can be used.

Let us now formalize the optimisation step. This step detects maximum of KD and groups all of the data objects in their neighbourhood into corresponding clusters. We use a hill climbing method for KD maxima estimation with Gaussian kernels (DENCLUE2) [9] and modify the technique for the multivariate case. This method converges towards a local maximum and adjusts the step size automatically at no additional costs.

Each KD maximum can be considered as the centre of a point cluster. With centre-defined clusters, every local maximum of $\hat{\Psi}(\cdot)$ corresponds to a cluster that includes all data objects that can be connected to the maximum by a continuous, uphill path in the function of $\hat{\Psi}(\cdot)$. Such centre-defined clusters allows for arbitrary-shaped clusters to be detected, including non-linear clusters. An arbitrary-shape cluster is the union of centre-defined clusters that have maximum that can be connected by a continuous, uphill path.

The goal of the hill climbing procedure is to maximize the KD $\hat{\psi}^{[\mathbf{X}]}(\mathbf{x})$. By setting the gradient $\nabla \hat{\psi}^{[\mathbf{X}]}(\mathbf{x})$ of KD to zero and solving the equation $\nabla \hat{\psi}^{[\mathbf{X}]}(\mathbf{x}) = 0$ for \mathbf{x} , we get:

$$\mathbf{x}^{(l+1)} = \frac{\sum_{\mathbf{x}_i \in \mathbf{X}} K_{\mathbf{H}}(\|\mathbf{x}^{(l)} - \mathbf{x}_i\|) \mathbf{x}_i}{\sum_{\mathbf{x}_i \in \mathbf{X}} K_{\mathbf{H}}(\|\mathbf{x}^{(l)} - \mathbf{x}_i\|)}. \quad (2)$$

The formula (2) can be interpreted as a normalized and weighted average of the data points. The weights for each data point depend on the influence of the corresponding kernels on $\mathbf{x}^{(l)}$. Hill climbing is initiated at each data point $\mathbf{x}_i \in \mathbf{X}$ and is iterated until the density does not change, i.e. $[\hat{\psi}^{[X]}(\mathbf{x}_i^{(l)}) - \hat{\psi}^{[X]}(\mathbf{x}_i^{(l-1)})] / \hat{\psi}^{[X]}(\mathbf{x}_i^{(l)}) \leq \epsilon$, where ϵ is a small constant. The end point of the hill climbing algorithm is denoted by $\mathbf{x}_i^* = \mathbf{x}_i^{(l)}$, corresponding to a local maximum of KD.

Now we should determine a cluster for \mathbf{x}_i . Let $\mathbf{X}^c = \{\mathbf{x}_1^c, \mathbf{x}_2^c, \dots\}$ be an ordered set of already identified cluster centres (initially, we suppose $\mathbf{X}^c = \emptyset$). First we find an index of the nearest cluster centre from \mathbf{x}_i^* in the set \mathbf{X}^c :

$$nc(\mathbf{x}_i^*) = \arg \min_{j: \mathbf{x}_j^c \in \mathbf{X}^c} \|\mathbf{x}_j^c - \mathbf{x}_i^*\|.$$

If the nearest cluster centre is close to \mathbf{x}_i^* , then the point \mathbf{x}_i is included in this cluster; otherwise, the point is used as a cluster centre to form a new cluster

$$\Lambda(\mathbf{x}_i) \leftarrow \begin{cases} nc(\mathbf{x}_i^*) & \text{if } \frac{\|\mathbf{x}_{nc(\mathbf{x}_i^*)}^c - \mathbf{x}_i^*\|}{\mathbf{x}_i^*} \leq \delta, \\ |\mathbf{X}^c| + 1 & \text{otherwise.} \end{cases}$$

where δ is a small constant and $\Lambda(x)$ is a class labeling function. In the second case, we also create a new cluster centre: $\mathbf{X}^c \leftarrow \mathbf{X}^c \cup \{\mathbf{x}_i^*\}$.

5 Decentralised KD-Based Clustering

In this section, we describe a cooperative method for sharing the clustering experience among agents in a network. While working with streaming data, one should take into account two main facts: (1) the nodes should coordinate their clustering experience over some previous sampling period, and (2) they must also adapt quickly to the changes in the streaming data without waiting for the next coordination action.

Let us first discuss the cooperation technique. Let $\mathbf{A} = \{A^j \mid 1 \leq j \leq p\}$ be a group of p agents. Each agent $A^j \in \mathbf{A}$ has a local dataset $\mathbf{D}^j = \{\mathbf{x}_t^j \mid t = 1, \dots, N^j\}$, $\mathbf{x}_t^j \in R^d$. To underscore the dependence of the KD function (1) from the local dataset of A^j , we denote the KD function by $\hat{\psi}^{[\mathbf{D}^j]}(\mathbf{x})$.

Consider a case when some agent A^i is unable to cluster a data point \mathbf{x}_t^i in some future time moment t because it does not have sufficient data nearby. By 'unable to cluster', we mean that this data point forms a new independent cluster after the optimisation step is performed. In this case, the agent A^i sends a request to other neighbour agents by sending the data point \mathbf{x}_t^i to them. Each agent A^j that receives the request tries to classify the point \mathbf{x}_t^i using its own KD function $\hat{\psi}^{[\mathbf{D}^j]}(\mathbf{x}_t^i)$ and performs an optimisation step to identify a cluster

for this point. If the optimisation step is successful, meaning that this point belongs to an existing cluster, the agent replies to A^i with information relevant to the neighbourhood of the requested point (parameters, artificial data points, random data points, etc.). Let us also define $\mathbf{G}^i \subset \mathbf{A}$, a group of agents that are able to reply to A^i with clustering information.

Our model uses a two-phase protocol for performing communication between agents. First, since A^i is unable to classify the data point \mathbf{x}_t^i , the information is sent to other agents. In response to the help-request, the neighbours A^j send parameters from their estimated KD functions. Since the KD function is non-parametric and estimated directly from observations, we approximate the function with a mixture of multi-dimensional Gaussian distributions. Agent A^j identifies cluster associated with point \mathbf{x}_t^i and performs the approximation of clusters with a mixture of normal distributions. Next, A^j transmits the cluster parameters (weight, mean and covariance matrix). The agent A^i adds this information to its KD and updates its clusters. Since parameter transmission requires less data, this approach requires less transmission, however, the approximation reduces the cluster shapes to a union of ellipsoids.

Let us consider an approximation step that approximates KD functions with a mixture of multivariate normal distributions. This step can be achieved with the expectation maximisation (EM) algorithm proposed by Dempster [4]. The approach is widely used for calculation of the maximum likelihood estimate of mixture models.

In a mixture model, the probability density function is

$$f(\mathbf{x}; \Theta) = \sum_{b=1}^B \pi_b f_b(\mathbf{x}; \Theta_b), \quad (3)$$

where π_b are positive mixing weights that sum to one, f_b are component density functions parameterized by Θ_b , and $\Theta = \{\pi_b, \Theta_b\}$ are the model parameters. Each observation is assumed to be from one of the B components. A common choice for component density is a multivariate normal distribution with parameters $\Theta_b = (\mu_b, \Sigma_b)$, where μ_b is a mean and Σ_b is a covariance matrix. Given a set of independent observations $\mathbf{X} = \{\mathbf{x}^1, \dots, \mathbf{x}^v\}$ in $\mathbf{x}^j \in \mathbf{R}^d$, the objective is to fit such a model to the data.

In the EM procedure, the expected likelihood of a given data-set is iteratively maximized. Let $\mathbf{z}^i \in \{0, 1\}^B$ be the membership indicator variable such that $z_b^i = 1$ if \mathbf{x}^i is generated by $f_b(\cdot)$ and 0 otherwise.

The E step simplifies to computing the conditional probabilities

$$\langle z_b^i \rangle = P \{z_b^i = 1 | \mathbf{x}^i; \Theta^{old}\} = \frac{\pi_b f_b(\mathbf{x}^i; \Theta^{old})}{\sum_l \pi_l f_l(\mathbf{x}^i; \Theta^{old})}. \quad (4)$$

In the M step, we have an update rule in closed form:

$$\hat{\pi}_b = \frac{1}{v} \sum_i \langle z_b^i \rangle, \quad \hat{\mu}_b = \frac{\sum_i \langle z_b^i \rangle \mathbf{x}^i}{\sum_i \langle z_b^i \rangle}, \quad (5)$$

$$\hat{\Sigma}_b = \frac{\sum_i \langle z_b^i \rangle (\mathbf{x}^i - \hat{\mu}_b)(\mathbf{x}^i - \hat{\mu}_b)^T}{\sum_i \langle z_b^i \rangle}. \quad (6)$$

The algorithm alternates between the E and the M steps until convergence is achieved.

We assume that each helping-agent $A^j \in \mathbf{G}^i$ receives data point \mathbf{x}_t^i and tries to classify it. If it is successful, A^j determines that \mathbf{x}_t^i belongs to a specific cluster and executes the EM-algorithm with this cluster. This algorithm approximates the cluster using a mixture of B^j multidimensional normal distributions with parameters $\Theta^j = \{\mu^j, \Sigma^j, \pi^j\}$, where $\mu^j = (\mu_1^j, \dots, \mu_{B^j}^j)$, $\Sigma^j = (\Sigma_1^j, \dots, \Sigma_{B^j}^j)$ and $\pi^j = (\pi_1^j, \dots, \pi_{B^j}^j)$, which are then returned to A^i .

After receiving all answers, the agent A^i has a vector of the parameters $\{\Theta^j\}$. The answers $\{\Theta^j\}$ can be interpreted by the agent A^i as data points μ^j with the only difference being that the additional weights π^j and bandwidths from Σ^j should now be taken into account. Denote $\hat{\mathbf{D}}^i$ as a dataset of the agent A^i that includes the received answers. Density estimates (1) of each agent are additive, i.e. the aggregated density estimate $\hat{\Psi}^{[\hat{\mathbf{D}}^i]}(\mathbf{x})$ can be decomposed into a sum of local density estimates and answers:

$$\hat{\Psi}^{[\hat{\mathbf{D}}^i]}(\mathbf{x}) = w_i \hat{\Psi}^{[\mathbf{D}^i]}(\mathbf{x}) + \frac{1 - w_i}{\sum_{\substack{A^j \in \mathbf{G}^i \\ b \in B^j}} \pi_b^j \cdot \sum_{A^j \in \mathbf{G}^i} B^j} \sum_{\substack{A^j \in \mathbf{G}^i \\ b \in B^j}} \pi_b^j K_{\Sigma_b^j}(\|\mathbf{x} - \mu_b^j\|), \quad (7)$$

where w_i is the weight assigned to own observations of the agent.

To measure the **clustering similarity** [3] among the agents $A^i \in \mathbf{A}$ we use the following representation of a class labeling by a matrix C with components:

$$C_{i,j} = \begin{cases} 1 & \text{if } x_i \text{ and } x_j \text{ belong to the same cluster and } i \neq j, \\ 0 & \text{otherwise.} \end{cases}$$

Let two labelings have matrix representations $C^{(1)}$ and $C^{(2)}$, respectively. We define a dot product that computes the number of pairs clustered together $\langle C^{(1)}, C^{(2)} \rangle = \sum_i \sum_j C_{i,j}^{(1)} C_{i,j}^{(2)}$. The Jaccard's similarity measure can be expressed as

$$J(C^{(1)}, C^{(2)}) = \frac{\langle C^{(1)}, C^{(2)} \rangle}{\langle C^{(1)}, C^{(1)} \rangle + \langle C^{(2)}, C^{(2)} \rangle - \langle C^{(1)}, C^{(2)} \rangle}. \quad (8)$$

6 Experimental Simulation Results and Case Studies

We simulated a traffic network in the southern part of Hanover (Germany). The network contained three parallel and five perpendicular streets, creating fifteen intersections with a flow of approximately 5000 vehicles per hour.

The service providers solved a travel time clustering problem using the factors listed in Table 1. They received information about the centrally estimated system variables (such as average speed, number of stops, congestion level, etc.) for this city district from TMS, combined it with their historical information, and made adjustments according to the information of other participants. They used the autonomous KD clustering (Section 4) and implemented the decentralised cooperative clustering algorithm (Section 5).

Table 1. System factors influencing the travel time

Variable	Description
Y	travel time (min);
X^1	length of the route (km);
X^2	average speed in the system (km/h);
X^3	average number of stops in the system (units/min);
X^4	congestion level of the flow (veh/h);
X^5	number of traffic lights in the route (units);
X^6	number of left turns in the route (units).

Note that the clustering of the factors $X^1 - X^6$ is performed together with the dependent variable Y . This clustering step can be considered as a pre-processing of initial data for the future forecasting with regression models. Different travel time Y forecasting models were obtained inside different clusters, that allows better fit of the models and better prediction considered in [5].

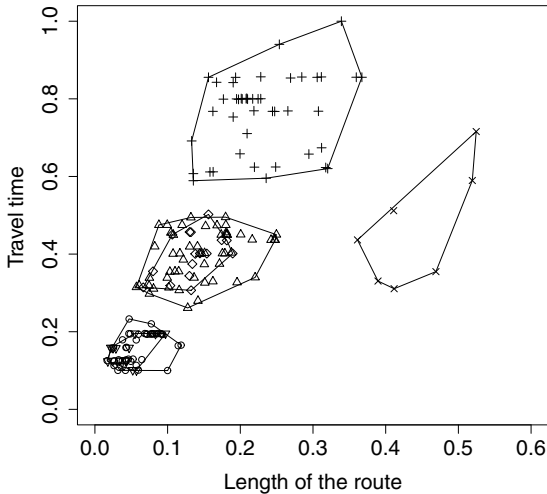


Fig. 2. A $X^1 - Y$ projection of 200 observations

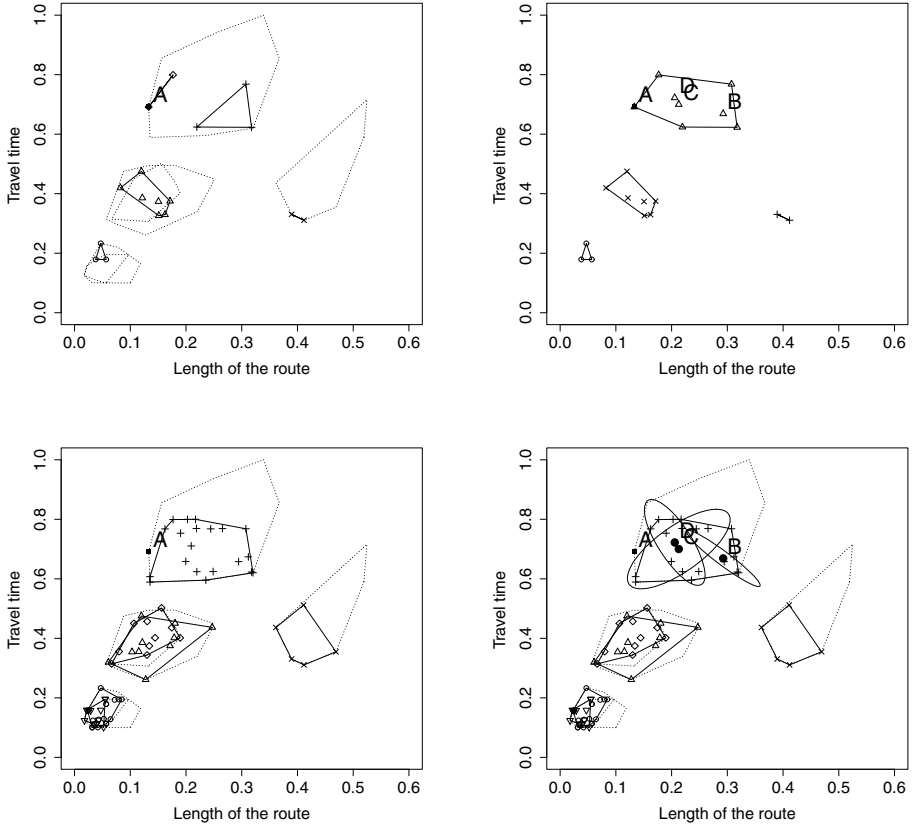


Fig. 3. A communication step between the requesting (top) and helping (bottom) agents

We simulated 10 agents with initial experience varied in range from 10 to 200 observations. Most simulation experiments ran for 200 time units. We assumed that all observations were homogeneous and the agents tried to estimate the same clusters. We normalized the initial data before the simulation execution.

To provide a visual overview of data, we presented the projection of 200 observations to the $X^1 - Y$ plane with the corresponding six clusters (Fig. 2). The visual intersection of the clusters in this projection is due to the difference of the point values in other dimensions.

For more accurate clustering, the agents used the presented decentralised cooperative KD clustering algorithm. Cooperation among the agents allowed improving clustering quality.

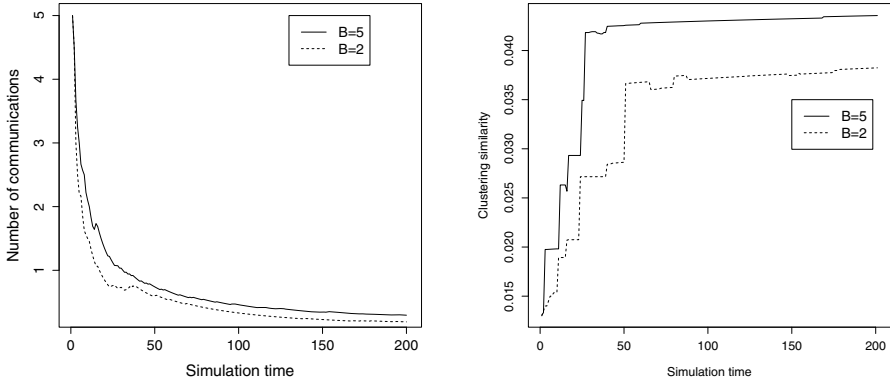


Fig. 4. A number of communication events (left) and similarity of agents' clusters (right) over time depending on B components in a mixture of multidimensional normal distributions

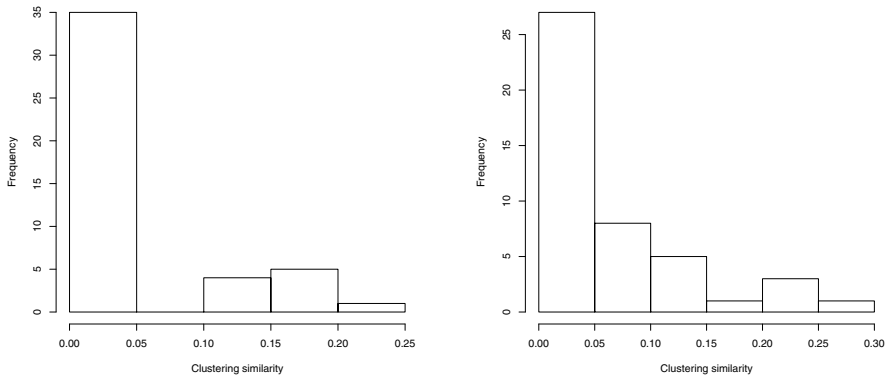


Fig. 5. Frequencies of clustering similarity for each pair of agents at the beginning (left) and at the end (right) of simulation

Let us illustrate one data synchronization step presented in Fig. 3. The requesting agent asks for help for point A (top left). The helping agent clustered the point using its own data, detected the corresponding cluster (bottom left), approximated it with the mixture of three normal distributions (shown as ellipses for two dimensional case with centres in B, C, D) and sent the corresponding parameters to the requesting agent (bottom right). The requesting agent added the obtained parameters as data points to its data and made new clustering (top right). This allowed to improve clustering similarity of these two agents

from 0.011 to 0.037 as well as clustering similarity of the requesting agent with an 'ideal' clustering from 0.004 to 0.006.

A system dynamics for a different number of transmitted points is shown in Fig. 4. Clustering similarity (right) increased faster for a bigger number of the estimated and transmitted components of normal distributions B^j for cluster approximations, but the number of communication events (left) decreased faster. Note, however, that one communication event was more expensive for a bigger number of transmitted points, but supplied more information.

Quality of the agent models was also checked by a cross-validation technique (Fig. 5) at the beginning (left) and at the end (right) of the simulation. These histograms shown a probability distribution of a similarity, which peak moved to bigger value after the coordination procedure.

7 Future Work and Conclusions

We developed the decentralised coordinated KD clustering approach for agent-based cloud computing architecture of intelligent transport system. The data coordination scheme is based on the transmission of parameters of multidimensional normal distribution, which approximate the cluster to which the requested point belongs. An experimental validation of the developed algorithm was also performed. Our future work is devoted to the development of new coordination schemes in proposed decentralised clustering approach.

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. *Communications of the ACM* 53(4), 50–58 (2010)
2. Bazzan, A.L.C., Klügla, F.: A review on agent-based technology for traffic and transportation. *The Knowledge Engineering Review FirstView*, 1–29 (2013)
3. Ben-Hur, A., Elisseeff, A., Guyon, I.: A stability based method for discovering structure in clustered data. In: *Pacific Sym. on Biocomputing*, vol. 7, pp. 6–17 (2002)
4. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Stat. Society. Series B* 39, 1–38 (1977)
5. Fiosina, J., Fiosins, M.: Chapter 1: Cooperative regression-based forecasting in distributed traffic networks. In: Memon, Q.A. (ed.) *Distributed Network Intelligence, Security and Applications*, pp. 3–37. CRC Press, Taylor and Francis Group (2013)
6. Fiosina, J., Fiosins, M., Müller, J.P.: Mining the traffic cloud: Data analysis and optimization strategies for cloud-based cooperative mobility management. In: Casillas, J., Martínez-López, F.J., Vicari, R., De la Prieta, F. (eds.) *Management Intelligent Systems. AISC*, vol. 220, pp. 25–32. Springer, Heidelberg (2013)
7. Fiosins, M., Fiosina, J., Müller, J.P., Görmer, J.: Reconciling strategic and tactical decision making in agent-oriented simulation of vehicles in urban traffic. In: *ICST Conf. on Simulation Tools and Techniques, SimuTools 2011* (2011)
8. Härdle, W., Müller, M., Sperlich, S., Werwatz, A.: *Nonparametric and Semiparametric Models*. Springer, Heidelberg (2004)

9. Hinneburg, A., Gabriel, H.H.: DENCLUE 2.0: Fast clustering based on kernel density estimation. In: Berthold, M., Shawe-Taylor, J., Lavrač, N. (eds.) IDA 2007. LNCS, vol. 4723, pp. 70–80. Springer, Heidelberg (2007)
10. Klusch, M., Lodi, S., Moro, G.: Agent-based distributed data mining: The KDEC scheme. In: Klusch, M., Bergamaschi, S., Edwards, P., Petta, P. (eds.) Intelligent Information Agents. LNCS (LNAI), vol. 2586, pp. 104–122. Springer, Heidelberg (2003)
11. Lee, J.G., Han, J., Whang, K.Y.: Trajectory clustering: A partition-and-group framework. In: ACM SIGMOD Int. Conf. on Management of Data (SIGMOD 2007), Beijing, pp. 593–604 (2007)
12. Li, Z.J., Chen, C., Wang, K.: Cloud computing for agent-based urban transportation systems. *IEEE Int. Systems* 26(1), 73–79 (2011)
13. Ogston, E., Overeinder, B., van Steen, M., Brazier, F.: A method for decentralized clustering in large multi-agent systems. In: Proc. of 2nd Int. Conf. on Autonomous Agents and Multiagent Systems, pp. 789–796 (2003)
14. Talia, D.: Cloud computing and software agents: Towards cloud intelligent services. In: Proc. of the 12th Workshop on Objects and Agents, vol. 741, pp. 2–6 (2011)
15. Weijermars, W., van Berkum, E.: Analyzing highway flow patterns using cluster analysis. In: Proc. of the 8th Int. IEEE Conf. on ITS, Vienna, pp. 831–836 (2005)

Evolving Mechanisms in Boolean Games

Cristiano Galafassi and Ana L.C. Bazzan

PPGC / Instituto de Informática, UFRGS
Caixa Postal 15064, CEP 91501-970, Porto Alegre, RS, Brazil
{cgalafassi,bazzan}@inf.ufrgs.br

Abstract. A Boolean game models situations in which each agent of a game holds a distinct set of Boolean variables, and has a goal it attempts to satisfy. However, at system level, there may be either constraints or a global goal to be fulfilled. Therefore, it is necessary to design a mechanism that provides incentives to the agents to align their individual goals with the global goal. It has been proven that designing such a mechanism is hard. Therefore, in this paper we propose the use of an evolutionary approach to mechanism design so that the system reward is optimized. This has a potential impact in distributed as well as multiagent systems, where agents often face binary decisions. Examples are minority and congestion games in general, as, e.g., the El Farol Bar Problem. Our results show that using a genetic algorithm one can evolve a configuration in which agents have Boolean functions that make them act in a way that is aligned with a global goal.

Keywords: Boolean games, Evolutionary mechanism design, Agent-based simulation, El Farol bar problem.

1 Introduction

In large scale distributed systems, it is important to understand whether individual agents' decisions can lead to globally optimal or at least acceptable solutions. One can approach this issue of aligning individual and global level goals either by designing the system already taking this issue into account, or by letting the system and/or its participants self-organize, the so-called automated mechanism design [5]. The idea is that instead of trying to design a mechanism that works for a range of settings only, one puts the computer to automatically compute the optimal mechanism for the specific setting at hand, e.g., by solving an optimization problem.

Our long term research aims at studying the effect of several types of strategies for self-organization of agents in complex systems. The present paper addresses simulation of agents' decision-making regarding a well-known problem in collectives in general [16] and in minority games in particular.

In the present paper we employ a scenario that is often used as a metaphor for coordination in social networks, the *El Farol* Bar Problem (EFBP), proposed by B. Arthur [1], with later contributions by, e.g., [3], [12], and [10]. The idea behind this metaphor is that a common situation people face is when one has

gone to his/her favorite pub only to find out that it happened to be overcrowded that night, leading to one regretting not to have stayed home.

We frame this problem as a Boolean game, a variant of random Boolean networks (RBNs). As explained in the next section, the problem of designing Boolean games is hard when not only local goals have to be satisfied, but also the global performance is to be observed. Usually this problem of designing mechanisms to given agents incentive in order to have a good global performance has been studied by economists, game theoreticians, and others. Despite the results, it remains a computationally hard problem. Therefore, we use an evolutionary approach to find good local rules that are aligned with the global goal.

The rest of this paper is organized as follows: The next section brings background material on related topics and reviews some works that relate to them. Following, in Section 3 our methods are presented. Simulations' results and their analysis then follow in Section 4. Section 5 discusses several aspects of this work and its future directions, and provide concluding remarks.

2 Background and Related Work

This section introduces and discusses background and related works in the following areas: coordination in multiagent systems, Boolean games, random Boolean networks, and evolutionary mechanism design.

2.1 Coordination in Multiagent Systems

Microeconomics and game-theory assume human behaviour to be rational and deductive – deriving a conclusion by perfect logical processes from well-defined premises. However, this assumption does not hold especially in interactive situations like the coordination of many agents. There is no *a priori* best strategy since the outcome of a game depends on other players. Therefore, bounded and inductive rationality (i.e., making a decision based on past experience) is supposed to be a more realistic description.

In this context, in 1994 Arthur introduced a coordination game called the *El Farol* Bar Problem. Every week N players wish to visit the bar *El Farol*. Up to a certain threshold ρ of customers the bar is very comfortable and enjoyable. If it is too crowded, it is better to stay home. The payoff of the game is clear: if the number of visitors is less than the threshold ρ , these visitors are rewarded, otherwise those who stayed home are better off. In the original work, $N = 100$ and $\rho = 60\%$ were used, but arbitrary values of N and ρ have also been studied, as, e.g., in [12].

Later, the EFBP was generalized to a binary (Boolean) game by [3], the so-called minority game (MG). An odd number N of players has to choose between two alternatives (e.g., yes or no, or simply 0 or 1). With a memory size m there are 2^{2^m} possible strategies. Each player has a set S of them. These are chosen randomly out of the whole set. In the simplest version of the game, players are

rewarded one point if they are in the minority group. Other functions that favor, for instance, smaller minorities were studied by several authors as, e.g., [4,12].

The MG and the EFBP are gradually becoming a paradigm for complex systems and have been recently studied in detail. We will refer briefly to some of the results.

In their original work, Challet and Zhang have systematically studied the influence of the memory size and number of strategies on the performance of the game. This and similar approaches do not consider any kind of structure of the network, i.e., how agents are connected.

In [10] and [6] a kind of social network was considered. However, in the former, the connectivity was such that the average number of neighbours with whom each agent interacts was fixed. In the latter, the connectivity degree K is not homogeneous (agents are connected based on preferential attachment). Another characteristic of [6] is that it was concerned with learning how to make optimal decisions at agent level, with a lesser focus on the global efficiency.

In more general terms, there has been an interesting line of research connecting minority games and collective intelligence such as [17]. For a discussion see [16].

2.2 Random Boolean Networks

Boolean networks have been used to explain self-organization and adaptation in complex systems. The study of the behaviour of regulatory systems by means of networks of Boolean functions was introduced by Kauffman in 1969 [13]. Examples of the use of this approach in biology, genomics, and other complex systems can be found in [14].

RBNs are made up of Boolean variables. A network of agents possessing such variables is composed of N agents that must decide which binary action to make. Variables in turn are, each, regulated by some other variables, which serve as inputs. Henceforth we use the term variable or agent indistinctly.

The dynamical behaviour of each agent, namely which action it will execute at the next time step, is governed by a logical rule based on a Boolean function. These functions specify, for each possible combination of K input values, the status of the regulated variable. Thus, being K the number of input variables regulating a given agent, since each of these inputs can be either on or off (1 or 0), the number of combinations of states of the K inputs is 2^K . For each of these combinations, a specific Boolean function must output either 1 or 0, thus the total number of Boolean functions over K inputs is 2^{2^K} .

To illustrate the regulation process, let us consider a simple example of a network of $N = 3$ agents where each was assigned a Boolean function randomly, and $K = 2$. The Boolean functions for these 3 agents are depicted in Table 1 (adapted from [14]): agents A and B are regulated by function OR, while agent C is regulated by an AND. In this table, one can see all possibilities for C (3rd column) to make a decision. Similarly, A's output is determined by the inputs from both B and C, and B's output depends on inputs from A and C.

Table 1. Boolean functions for agents C, A and B

(AND)			(OR)			(OR)		
A	B	C	B	C	A	A	C	B
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	1

Given the three Boolean functions that appear in Table 1, Table 2 shows, for all 2^3 states at a given time T , the action taken by each agent at time $T + 1$, i.e., the successor state of each state. Further, from this table, it is possible to determine the state transition graph of the network, which appears in Figure 1. One sees that there are 3 state cycles (attractors).

Table 2. States' transition for Table 1

(T)			(T+1)		
A	B	C	A	B	C
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	1	1	1

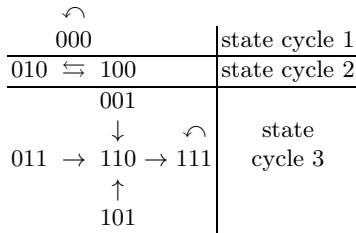


Fig. 1. States' transition graph for Table 1 (3 state cycles and attractors)

If we randomly assign one of the 2^{2^K} Boolean functions to each of the N agents, the dynamics is deterministic and the system ends up in one of the state cycles. Depending on the application domain, some of these states may be undesirable. For instance, in minority games, in both cycles 1 (000) and 3 (111), either none (state 1) or all (state 3) select a given action. Clearly, this is not a good strategy to play a minority game.

Table 3. Mutated version of Table 1

(NAND)			(OR)			(OR)		
A	B	C	B	C	A	A	C	B
0	0	1	0	0	0	0	0	0
0	1	1	0	1	1	0	1	1
1	0	1	1	0	1	1	0	1
1	1	0	1	1	1	1	1	1

Table 4. States' transition for Table 3

(T)			(T+1)		
A	B	C	A	B	C
0	0	0	0	0	1
0	0	1	1	1	1
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	1	1	0

However, if the topology of the network of the just mentioned example evolves, then the system may escape a bad attractor (again, from the point of view of a minority game), which is a state where either too many or too few agents make a given action. This evolution of the network may happen in several ways: agents get reconnected to others, the Boolean functions change, etc.

Let us consider an example in which the Boolean function of agent C changes from AND to NAND. Functions are now depicted in Table 3 while Table 4 shows the successor state of each state and Figure 2 depicts the state transition graph. The dynamics of the regulation changes as seen in Figure 2. Now only one state or attractor exists (110), namely one that has the property that agents A and B are always in the majority while agent C is in the minority.

The extent of such a change – whether or not the system will be attracted to another attraction basin – obviously depends on the extent of the changes in the network and/or functions. In [14] the author extensively discusses many of these factors, as well as the properties of RBNs, including the issue of stability and cycle length. In the present paper, because the logic of the functions and the structure of the network changes along time, properties such as periodic behaviour cannot be observed.

On the other hand, a central question raised by Kauffman, which is relevant to our work, relates to the problem of adaptation in a system with many interacting parts. The key question is whether an adaptive process which is constrained to altering the input connections between elements in a network and the logic governing them can hill climb to networks with desired attractors.

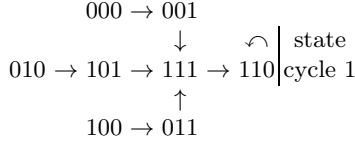


Fig. 2. States’ transition graph for Table 3 (single state cycle and attractor)

2.3 Boolean Games

Boolean games model situations in which each agent holds a distinct set of Boolean variables, and has a goal it attempts to satisfy. An agent’s goal is represented as a propositional logic formula over a set of Boolean variables, where some of these variables have values that are not necessarily set by the agent. The actions that an agent can take consist of assigning values to the variables it holds.

As in [11,8,9] (with slight variants), a Boolean game is a $2n + 3$ tuple:

$$G = \langle N, \Phi, c, \gamma_1, \dots, \gamma_n, \varphi_1, \dots, \varphi_n \rangle$$

where $N = \{1, \dots, n\}$ is the set of agents, $\Phi = \{p, q, r, \dots\}$ is a finite set of Boolean variables, $c : \Phi \times B \rightarrow R_{\geq}$ is a cost function for available assignments, $\gamma_1, \dots, \gamma_n$ are the goals of the N agents, and $\varphi_1, \dots, \varphi_n$ is a partition of variables Φ over the agents.

The aim of each agent N_i is to choose an assignment to the variables φ_i it controls, so as to satisfy its goal γ_i . As mentioned, an issue is that γ_i may contain variables controlled by other agents who are also trying to choose values for their variables to get their goals satisfied as well.

In the formulation we follow, Boolean variables are, each, controlled by one agent N_i , with φ_i being the set of variables controlled by agent N_i . Controlling variables means that the agent N_i has the ability to set the values for each variable $p \in \varphi_i$. Each agent can choose an allocation of truth or falsity to φ_i .

An outcome $(v_1, \dots, v_n) \in V_1 \times \dots \times V_n$ of the game is a collection of choices, one for each agent.

In this particular paper, we are not focusing on optimization at agents level. Rather, we are interested in finding an assignment of Boolean functions that optimizes the global level. This is the task of mechanism design, i.e., finding an assignment of truth and falsity for each Boolean variable that is globally efficient. Therefore, we set all costs to value $c = 1$. This means costs play no role in the Boolean game.

However, even the problem of finding such an assignment is hard. As shown in [2] and [7], Boolean games are computationally complex. Therefore, in order to apply this formalism, this issue of computational complexity must be treated. In the present paper we aim at using an evolutionary approach to design a mechanism that assigns each agent a Boolean function so that the global goal is reached.

2.4 Evolutionary Mechanism Design

Mechanism design deals with designing rules of a game to achieve a specific outcome. In such a design process, the designer attempts “to achieve socially-desirable outcomes despite the impossibility of prescribing the exact behaviour of the market participants, who may attempt to subvert the market for their own personal gain” [15].

Given the size of problems in large-scale, open and distributed systems, some mechanisms may yield suboptimal solutions, which is not a desirable outcome. In an attempt to deal with this situation, an approach to designing mechanisms has been proposed, which is based on the principle of satisfying instead of optimizing: evolutionary mechanism design [15]. We remark that this is a form of *automated* mechanism design, given that it is done automatically by a computational method. This approach considers that the agent will decide optimally upon its neighbours decision, but there is no guarantees that the method converges to an equilibrium.

In this approach, small changes in the underlying rules of the game can have big effects on the behaviour of the agents using these rules, and can alter the dynamics of the system.

In the present paper we employ a genetic algorithm to evolve populations of networks of agents connected in a regular topology, where each agent has a Boolean function that regulates its decision-making. As explained before, such function takes other agents past decisions as input. In this way we model a social network of decision-makers as a Boolean game, whose aim is to perform or not an action (here, to go to the bar) depending on what the acquaintances have done. However, in the case of the bar, because it has a certain capacity (a global constraint), we want to design agents and functions so that this global goal is also considered.

3 Methods

This section discusses the methods used. We start with the underlying methods used to equip agents with a decision-making device that is based on the concept of Boolean games and RBNs. Following, in Section 3.2 and Section 3.3 we introduce both the model without and with the evolutionary approach respectively.

3.1 Decision-Making Using a Boolean Function

The decision-making framework used here is appropriate for binary (i.e., Boolean) decision-making, which aims at considering inputs from other agents in the decision-making process of each agent. To do so we replace the space of possible strategies described in [1] by a set (one per node) of Boolean functions. This also means that each node is connected to a given number of others.

Agents have to make decisions regarding a binary action. In particular, our scenario is based on the EFBP of [1]. Thus the decision is about whether or not to

go to a bar. Contrarily to the work in [1] (and many others that have followed), we consider that the agents are organized in a kind of social network. Thus instead of having a random strategy, each node has random Boolean functions and uses them to determinate whether or not to go to the bar. In this social network, a decision is based on the previous behaviours of each agent's acquaintances. The rationale is that agents want to go to the bar if others go as well. Now, there are two types of agents in the society. Some are very concerned with going only if the whole group of friends go. Here we consider a low number of friends, as explained next. Thus, this assumption is reasonable. The other type of agent is the one that goes to the bar if at least one of its friends also goes. Henceforth we call this types of agents a_{\wedge} and a_{\vee} respectively.

From the discussion of the RBN example given in Section 2.2, we remark that if no mechanism is designed to direct agents to the global goal, then, because of the Boolean functions assigned randomly, the system may end up in an undesirable attractor. In an automated mechanism design framework however, agents with suitable Boolean functions may evolve, thus giving incentives to agents to behave in a given way. Therefore, the study of which configurations of the network (here, in terms of Boolean functions) are favourable, is key. Also as mentioned, it is impracticable to do this task analytically for a high number of agents. Thus the evolutionary approach proposed here.

In the next subsection we detail the basis of the Boolean game that represents the situation of agents trying to decide whether or not to go to the bar with their acquaintances. Then, we give the details of the evolutionary approach.

3.2 Basic Boolean Game

The model used here can be described as follows. N agents are part of a regular network, where each has exactly K acquaintances.

The decision-making of agent i is guided by a Boolean function $f_i \in \mathcal{F}$. The inputs to f_i are the decision made by the agent acquaintances in the last time step. Depending on these decisions and on f_i , agent i decides whether or not to go to the bar at each time step.

The game is played for t_{max} time steps. We are interest in the number of agents that decide to cooperate, expressed by $Q_{(t)}$. The interaction process is shown on Algorithm 1. This means that in each time step, each agent performs the decision-making as shown there.

After the main parameters are set (lines 1–2), each agent is connected to other K agents and gets one Boolean function assigned randomly (line 3); the previous action is also randomly assigned. The following is then repeated for t_{max} time steps: each agent i checks the actions of its K acquaintances in the last time step. These actions are inputs to i 's Boolean function so that i 's action for the current time step is determined (line 9). At each time step each agent uses its Boolean function $f_i \in \mathcal{F}$. In the present paper, \mathcal{F} has only two possibilities: either \wedge or \vee .

According to f_i and also to the value of the K_i entries, either 0 or 1 is output. It must be noticed that the inputs are the actions of the K neighbors in the

Algorithm 1. Basic model for deciding whether or not to go to the bar

```

1: INPUT: global variable  $t_{max}$  //max. number of games
2: INPUT:  $N$  // number of agents
3: assigns agent's acquaintances, random Boolean functions and random initial decisions
4: while not  $t_{max}$  do
5:   for each agent  $i$  do
6:     for each acquaintance  $j$  do
7:       get input of  $j$  // gets decision of  $j$  in the last time step
8:     end for
9:     decide whether to go to the bar according to  $i$ 's Boolean function and  $j$ 's inputs
10:   end for
11: end while
12: compute  $Q_{(t)}$ 

```

last time step (in the initial time step, all agents start with a random action). In the bar scenario we assume that if a Boolean function returns 0, this means the agent stays at home; when the function returns 1 the decision is to go to the bar.

At the end (line 12 of Algorithm 1) the bar attendance $Q_{(t)}$ is computed. We use $Q_{(t)}$ as a measure of efficiency of the game. Efficiency is a domain-dependent concept related to the equilibrium of the particular system. In the bar scenario, the equilibrium calls for the bar accommodating ρ agents (as also in the original work of [1], where $\rho = 60\%$). Thus, we are interested in a mechanism which $\rho\%$ of the agents go to the bar.

Algorithm 2. Evolutionary approach

```

1: INPUT: global variable  $t_{max}$  //max. number of games
2: INPUT:  $N$  //number of agents
3: INPUT:  $|S|$  //population size
4: INPUT:  $p_{mut}$  //mutation probability
5: INPUT:  $|e|$  //elite size
6: INPUT:  $g_{max}$  //number of generations
7: INPUT:  $\rho$  //number of desirable agents in bar
8: create a population of  $|S|$  individuals/networks
9: assigns agent's acquaintances, random Boolean functions and random initial decisions
10: while not  $g_{max}$  do
11:   for each individual in the population do
12:     while not  $t_{max}$  do
13:       for each agent  $i$  do
14:         for each acquaintance of  $j$  do
15:           get input of  $j$  // gets decision of  $j$  in the last time step
16:         end for
17:         decide whether to go to the bar according to  $i$ 's Boolean function and  $j$ 's inputs
18:       end for
19:     end while
20:     compute fitness  $\eta$  and  $Q_{(t)}$ 
21:   end for
22:   create a new population including a copy of the  $|e|$  best individuals
23:   while new population not complete do
24:     using roulette wheel select individual according to its fitness and mutate each agent with
       probability  $p_{mut}$ 
25:   end while
26: end while
27: compute  $\langle Q_{(t)} \rangle$ 

```

3.3 Evolutionary Approach

To obtain a certain bar attendance, it is necessary to design or evolve a Boolean network with a specific configuration. In order to try to obtain this particular configuration, a genetic algorithm (GA) is used. Here we remark that, keeping GA terminology, we use the word individual to represent an individual in the GA population. This means that an individual is a network of agents, each connected with K others, making decisions about whether or not to go to the bar. Thus an individual is a set of N agents connected with K acquaintances, where each agent possesses a Boolean function f_i .

$|S|$ denotes the population size. Each agent of the selected individual mutates to a different function $f_i \in \mathcal{F}$ with probability p_{mut} . This can be seen as a metaphor for agents performing experimentation (i.e., being more or less restrict in what concerns going to the bar with friends). Finally, we denote the maximum number of generations as g_{max} .

The fitness function aims at the global goal, i.e., having ρ agents in the bar, and is shown in Equation 1.

$$\eta = N - (abs(Q_{(t)} - \rho)) \quad (1)$$

In Equation 1, N represents the total number of agents on the network, $Q_{(t)}$ represents the number of agents who decide to go to the bar at time step t , and ρ is the ideal number of agents in the bar.

Algorithm 2 shows how the genetic algorithm is used. Initially, all parameters are set (lines 1-7). After that, the population is created randomly (line 8) and each agent is connected to other K agents; each agent is assigned a Boolean function and a previous decision randomly (line 9). The evolutionary model evaluates the stop criterion, g_{max} (line 10). The algorithm then continues basically as in Algorithm 1 for the whole population of networks. The difference is that now it includes the computation of the fitness η , besides the bar attendance $Q_{(t)}$ (lines 11-21).

Next, a new population is generated with the $|e|$ best individuals of the current population (line 22), where the rest of the agents are chosen by a roulette wheel method in proportion to η . Mutation is then applied with rate p_{mut} . At the end (line 27) we compute $\langle Q_{(t)} \rangle$ which represents the average of $Q_{(t)}$ over all individuals of the population. $\langle Q_{(t)} \rangle$ is the metric used to evaluate the quality of the evolutionary model in terms of bar attendance. We remark that this metric refers to the *average value of the population*. We avoid using the best individual because it is generally the case that already the initial population does contain a network with a relatively good value for $Q_{(t)}$.

4 Experiments and Analysis of the Results

4.1 Settings

For the experiments discussed next, the parameters previously mentioned in Section 3 take the values as in Table 5.

Table 5. Parameters and their values

Parameter	Value
N	121
K	4
t_{max}	10
$ \mathcal{F} $	2
ρ	$\approx 25\%$ (30 agents)
p_{mut}	variable

Agents are created so that the number of a_\wedge (henceforth N_\wedge) as well as the number of a_\vee (henceforth N_\vee) is around 50%.

Regarding the number of times steps decisions are made by the agents (about whether or not to go to the bar), in this paper we use $t_{max}=10$, which was determined experimentally: it was observed that normally 5 time steps were necessary for the model to reach an attractor state.

As said, the number of Boolean functions available to each agent is $|\mathcal{F}| = 2$: either \wedge or \vee . The main metric to be analysed is the average amount of agents that go to the bar, considering all individuals in the population ($\langle Q_{(t)} \rangle$). In this experiment we use a grid with size 11x11 (i.e. $N = 121$), where each agent is connect to 4 neighbours ($K = 4$). $\rho \approx 25\%$ of the agents.

4.2 Results

Next we show the results of experiments using no evolutionary process, i.e., the method described in Algorithm 1 was used. This results discussed next refer to 100 repetitions of that algorithm.

Table 6 shows, after $t_{max} = 10$ time steps (i.e., after the attractor was reached): average number of agents using each Boolean function (N_\wedge and N_\vee), their standard deviations, and the average and standard deviation of bar attendance $Q_{(t)}$

Table 6. Basic Boolean game model: number of agents using functions \vee and \wedge , and bar attendance

	Average	Standard Deviation
N_\vee	60.69	4.88
N_\wedge	60.31	4.88
$Q_{(t)}$	60.13	8.22

It can be seen in Table 6 that the Boolean functions are equally distributed. The same happens with the number of agents that go to the bar: around 50% of the $N = 121$ decide to go.

As additional information, considering all 100 repetitions, the maximum N_\vee was 81 and the maximum N_\wedge was 79. In the same way, the maximum number of agents going to the bar was 99 and the lowest was 40.

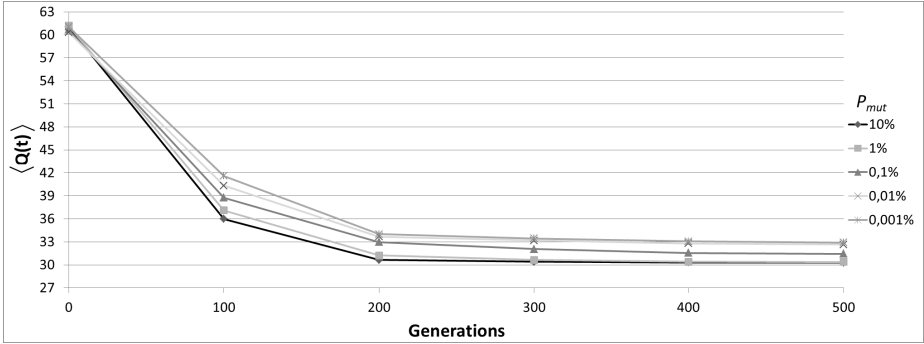


Fig. 3. $\langle Q(t) \rangle$ as a function of the number of generations

As expected, the global goal of ρ agents going to the bar was not achieved using a random distribution of functions among the N agents. To deal with this, the evolutionary approach presented in Algorithm 2 was then applied, whose results are shown next. Here, the algorithm was run varying $p_{mut} \in \{10\%, 1\%, 0.1\%, 0.01\%, 0.001\%\}$. Figure 3 shows the results for each case. The vertical axis indicates the average number of agents that decided to go to the bar ($\langle Q(t) \rangle$). Results are average values for 100 runs with each p_{mut} value.

One notices that, regarding the results in Table 6, there is an improvement along the generations. Figure 3 shows that $\langle Q(t) \rangle$ decreases as the population evolves, especially in the first 200 generations. When $p_{mut}=0.01\%$, the difference between $\langle Q(t) \rangle$ and ρ was only about 2 agents, given that the target is to have $\langle Q(t) \rangle = 30$.

As additional information, considering 100 experiments with each p_{mut} , when $p_{mut}=10\%$, in all runs $\langle Q(t) \rangle$ has converged to ρ . This was also the case with $p_{mut}=1\%$. When $p_{mut}=0.1\%$, 75 experiments had $\langle Q(t) \rangle$ converging exactly to ρ ; with $p_{mut}=0.01\%$, $\langle Q(t) \rangle$ converged to ρ in 28 experiments; and when $p_{mut}=0.001\%$, only 11 converged.

Table 7 shows $\langle Q(t) \rangle$, N_{\vee} , and N_{\wedge} at the end of e, as well as their standard deviations for the experiments with the evolutionary approach. It can be verified that the evolutionary approach has reached the global goal, obtaining the expected bar attendance. Besides, this quantity is associated with a lower standard deviation, when compared to the non-evolutionary approach.

Table 7. Evolutionary Model: N_{\vee} , N_{\wedge} , and $\langle Q(t) \rangle$ (values at last generation, average over 100 repetitions)

Attributes	10%		1%		0.1%		0.01%		0.001%	
	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.	Avg.	St. Dev.
N_{\vee}	36.38	2.30	36.38	2.33	37.24	2.79	37.43	2.92	37.71	3.15
N_{\wedge}	84.62	2.30	84.62	2.33	83.76	2.79	83.57	2.92	83.29	3.15
$\langle Q(t) \rangle$	30.00	0.00	30.00	0.00	30.29	1.35	32.19	2.12	32.15	2.16

5 Conclusion and Future Work

One of the research directions in adaptive and self-organizing systems is dedicated to how to design agents that are able to coordinate decisions and actions. This can be done manually or in an automated way, in which a mechanism is either learned or evolves so that agents get an incentive to align their private goals to the system goal.

This paper has presented an evolutionary approach for finding a mechanism that works on a Boolean game. In particular, we use a scenario in which agents in a social network have to decide whether or not to go to a bar giving the decisions of their acquaintances. These agents have a private goal given by their Boolean functions (go to the bar only if at least one friend goes, or go to the bar only if all friends go). However, there is a global constraint that relates to an optimal bar attendance.

It was shown that with a decision-making process based on Boolean functions regulating agents binary decisions, agents do fulfill their private goals but the bar gets overcrowded since nearly twice as many agents go to the bar.

In addition, the deviation over the number of agents using each Boolean function is high due to the high number of attractor states that underlie this dynamic system, a clear combinatorial problem.

To deal with this, an evolutionary approach based on genetic algorithms was proposed. This way, mutations were introduced in the system (a metaphor for experimentation at agents level), which had led the overall system to satisfy the global constraint: only about 30 agents decide to go to the bar. Moreover, when $p_{mut}=10\%$ and $p_{mut}=1\%$, in all 100 repetitions of the experiments, the bar attendance has converged to ρ (the ideal attendance).

As future work we suggest the application of this technique in different kinds of network topologies such as scale free ones, as well as the experimentation with other Boolean functions. In the case of different Boolean functions, it would be interesting to base the distribution of such functions in findings from a sociological study.

References

1. Arthur, W.B.: Inductive reasoning and bounded rationality. *The American Economic Review* 84(2), 406–411 (1994)
2. Bonzon, E., Lagasque-Schiex, M.-C., Lang, J., Zanuttini, B.: Boolean games revisited. In: *ECAI*, pp. 265–269 (2006)
3. Challet, D., Zhang, Y.C.: Emergence of cooperation and organization in an evolutionary game. *Physica A* 246, 407–418 (1997)
4. Challet, D., Zhang, Y.C.: On the minority game: Analytical and numerical studies. *Physica A* 256, 514–532 (1998)
5. Conitzer, V., Sandholm, T.: Complexity of mechanism design. In: *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence, UAI 2002*, pp. 103–110. Morgan Kaufmann Publishers Inc., San Francisco (2002)

6. Bazzan, A.L.C., Epstein, D., Machado, A.M.: Break with agents who listen to too many others (at least when making Boolean decisions!). In: Proceedings of the Eleventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012), pp. 1389–1390 (2012)
7. Dunne, P.E., Wooldridge, M.: Towards tractable Boolean games. In: AAMAS, pp. 939–946 (2012)
8. Endriss, U., Kraus, S., Lang, J., Wooldridge, M.: Designing incentives for Boolean games. In: AAMAS, pp. 79–86 (2011)
9. Endriss, U., Kraus, S., Lang, J., Wooldridge, M.: Incentive engineering for Boolean games. In: IJCAI, pp. 2602–2607 (2011)
10. Galstyan, A., Kolar, S., Lerman, K.: Resource allocation games with changing resource capacities. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 145–152. ACM Press (2003)
11. Harrenstein, P., van der Hoek, W., Meyer, J.-J., Witteveen, C.: Boolean games. In: Proceedings of the Eighth Conference on Theoretical Aspects of Rationality and Knowledge, pp. 287–298. Morgan Kaufmann Publishers, San Mateo (2001)
12. Johnson, N.F., Jarvis, S., Jonson, R., Cheung, P., Kwong, Y.R., Hui, P.M.: Volatility and agent adaptability in a selforganizing market. *Physica A* (258), 230–236 (1998)
13. Kauffman, S.A.: Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol.* 22(3), 437–467 (1969)
14. Kauffman, S.A.: *The Origins of Order*. Oxford University Press, Oxford (1993)
15. Phelps, S., Mcburney, P., Parsons, S.: Evolutionary mechanism design: a review. *Autonomous Agents and Multi-Agent Systems* 21(2), 237–264 (2010)
16. Tumer, K., Wolpert, D.: A survey of collectives. In: Tumer, K., Wolpert, D. (eds.) *Collectives and the Design of Complex Systems*, pp. 1–42. Springer (2004)
17. Wolpert, D.H., Wheeler, K., Tumer, K.: Collective intelligence for control of distributed dynamical systems. *Europhysics Letters* 49(6) (March 2000)

Repeated Sequential Single-Cluster Auctions with Dynamic Tasks for Multi-Robot Task Allocation with Pickup and Delivery

Bradford Heap and Maurice Pagnucco

School of Computer Science and Engineering
The University of New South Wales
Sydney, NSW, 2052, Australia
{bradfordh,morri}@cse.unsw.edu.au

Abstract. In this paper we study an extension of the multi-robot task allocation problem for online tasks requiring pickup and delivery. We extend our previous work on sequential single-cluster auctions to handle this more complex task allocation problem. Our empirical experiments analyse this technique in the domain of an environment with dynamic task insertion. We consider the trade-off between solution quality and overall planning time in globally reallocating all uncompleted tasks versus local replanning upon the insertion of a new task. Our key result shows that global reallocation of all uncompleted tasks outperforms local replanning in minimising robot path distances.

1 Introduction

Consider a team of autonomous mobile robots operating as courier delivery vehicles in a large office-like environment (Fig. 1). Each robot is required to pickup from and deliver parcels to a variety of locations around the office building. Each robot may be constrained to a fixed capacity in the number of parcels it can carry at any one time and, after a parcel is picked up, it can only be delivered to its intended destination. Our goal is for the robots to be allocated and deliver all parcels as effectively and efficiently as possible.

The task allocation problem with pickup and delivery is an extension of the widely studied *multi-robot task allocation (MRTA) problem* which, in general, considers each task as a single location to visit. There are many existing approaches for solving this class of problem. However, most existing techniques require that all tasks are static and known before allocation. In many real-world situations, additional tasks are dynamically discovered during execution.

In this paper we extend our previous work on solving MRTA problems through auctioning clusters of geographically close tasks [6]. We propose an algorithm for the formation of task clusters based on pickup and delivery locations. We apply this extended technique to a scenario with dynamically inserted tasks. We compare these results to a scenario where all tasks are known at the outset. Finally, we consider the trade-off between solution quality and overall planning

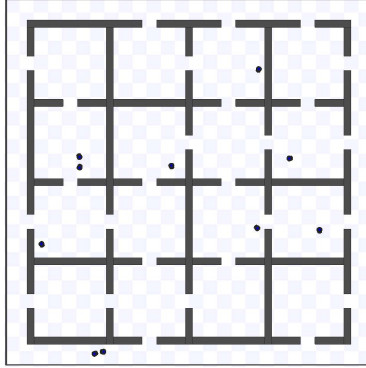


Fig. 1. A simulation of robots operating in an office-like environment

time in globally reallocating all uncompleted tasks versus local replanning when a robot is assigned a previously unknown task.

2 Problem Definition

Multi-robot routing is considered the standard testbed for MRTA problems [2]. We expand the problem formalisation given by Koenig *et al.* [11] to include tasks with pickup and delivery. Given a set of robots $R = \{r_1, \dots, r_m\}$ and a set of tasks $T = \{t_1, \dots, t_n\}$. A partial solution to the MRTA problem is given by any tuple $\langle T_{r_1}, \dots, T_{r_m} \rangle$ of pairwise disjoint task subsets:

$$T_{r_i} \subseteq T \text{ with } T_{r_i} \cap T_{r_{i'}} = \emptyset, i \neq i', \forall i = 1, \dots, m \quad (1)$$

Each task subset T_{r_i} is then assigned to a single robot $r_i \in R$. To determine a complete solution we need to find a partial solution where all tasks are assigned to task subsets:

$$\langle T_{r_1} \dots T_{r_m} \rangle \text{ with } \cup_{r_i \in R} T_{r_i} = T \quad (2)$$

For tasks with pickup and delivery, the structure of each task t is a tuple $t = \langle l_p, l_d \rangle$ of a pickup location l_p and a delivery location l_d . We consider a robot to be executing a task once it has visited its pickup location and until the point it has reached its delivery location. Robots may have capacity constraints in the number of tasks they are able to execute at any moment in time. This is representative of real robots which may have a fixed maximum number of items they can carry.

We assume robots have perfect localisation¹ and can calculate the cost λ to travel between locations. The cost to travel between any two locations is equal across all robots. The robot cost $\lambda_{r_i}(T_{r_i})$ is the minimum cost for an individual

¹ This allows us to focus on the effectiveness of the bidding method.

robot r_i to visit all locations T_{r_i} assigned to it. There can be synergies between tasks assigned to the same robot, such that:

$$\lambda_{r_i}(\{t\}) + \lambda_{r_i}(\{t'\}) \neq \lambda_{r_i}(\{t\} \cup \{t'\}) \quad (3)$$

A positive synergy is when the combined cost for a robot to complete two tasks is lower than the individual costs for the robot to complete each task:

$$\lambda_{r_i}(\{t\} \cup \{t'\}) < \lambda_{r_i}(\{t\}) + \lambda_{r_i}(\{t'\}) \quad (4)$$

Generating a valid solution to the standard MRTA problem is not difficult. For instance, a simple approach is to assign each task in turn to a randomly selected robot. However, this approach gives no guarantees on the execution time, energy or resources used in completing the assigned tasks. Subsequently, the application of *team objectives* arises to provide additional guidance in the search for solutions to the task allocation that meet certain criteria. For instance, some common desires of a multi-robot system are minimising time spent in execution of tasks, minimising energy or fuel consumed, and/or even distribution of tasks across all robots.

Lagoudakis *et al.* discusses team objectives in detail and their application to MRTA [13]. In this work we use two commonly adopted team objectives:

MiniMax $\min \max_{r_i \in R} \lambda_{r_i}(T_{r_i})$ that is, to minimise the maximum distance any individual robot travels.

MiniSum $\min \sum_{r_i \in R} \lambda_{r_i}(T_{r_i})$ that is, to minimise the sum of the paths of all robots in visiting all their assigned locations.

The application of these two team objectives to solving the MRTA problem can generate vastly different allocations of tasks to robots. The MiniMax team objective can be considered as the Min-Max Vehicle Routing Problem or the Makespan problem and the MiniSum team objective can be considered as a multi-robot version of the Travelling Salesperson Problem [22].

3 Related Work

3.1 Market-Based Task Allocation

Market-based distributed auction algorithms are popular in the robotics community for solving standard MRTA problems [2,9]. An optimal allocation of tasks to robots can be determined using a single-round *combinatorial auction* [1]. However, winner determination in most combinatorial auctions is NP-complete; they have high communication costs, and are therefore very slow and generally not used in practical applications.

An alternative approach is *sequential single-item auctions* (SSI auctions) which allocate tasks over multiple rounds [13,10]. Despite SSI auctions producing team costs that are generally sub-optimal, they have much lower communication and winner determination costs which result in a quicker allocation of tasks. A variety of improvements and extensions to SSI auctions have been studied which trade off allocation time against overall team costs [9].

A major weakness in sub-optimal auction algorithms with one-shot task assignments is their inability to avoid local minima. For instance, consider the bidding rules for the MiniSum and MiniMax team objectives in standard SSI auctions. When no tasks are assigned to an individual robot, the robot will always bid for the task closest to it which, in many situations, will not be an optimal task assignment. Once they have been assigned this task, all subsequent bids will factor in the inter-task synergies of this task and, as a result, solutions that are far from optimal are developed [18].

To refine a task allocation solution post-initial allocation in SSI auctions Nanjanath and Gini switch the bidding rule from MiniSum to a MiniMax like approach [16]. Adapting their previous re-auction approach [15] the initial allocation of tasks to robots is done using a standard SSI auction. Then, upon each task completion, all uncompleted tasks are auctioned under the requirement of minimising execution time. Robots only exchange tasks if it improves the overall team objective and, once a task has been exchanged, the robots involved in the exchange replan their paths to travel.

An alternative approach to improving the solution to SSI auction solution with capacity constraints is with K -swaps [25]. K -swaps generalise Sandholm's previous work on contracts for task exchanges [17] and allow two or more robots to exchange differing numbers of tasks to reduce the overall team cost. K -Swaps has been experimentally shown to make significant improvements to task allocations, however, the algorithm trades off larger improvements with pronouncedly increased computational time. Furthermore, task swaps and transfers can also be considered with auctions that incorporate simulated annealing to avoid local minima [24]. Such approach randomly selects tasks and robots for task exchanges and calculates probabilities of the task exchange being accepted and over time can generate an optimal solution.

3.2 Task Clustering

A major computational challenge in the performance of auction mechanisms that consider inter-task synergies is their ability to handle large numbers of robots and tasks. In many auction algorithms increasing the number of tasks causes a combinatorial explosion in the number of calculations required to form task bids. Further compounding this, as the number of robots increases, the communication and computational requirements for winner determination also increase. As a result the suitability of these techniques in large real-world scenarios is limited.

Forming clusters of tasks has been explored by a number of researchers as a method to reduce the combinatorial explosion of increasing task counts. In early work on market-based task allocation Sandholm expanded the *Contract Net Protocol* (CNP) [20] by introducing *C-contracts* which replace the CNP's standard one item contract with a contract for a cluster of tasks all of which the contracted robot must complete. Sandholm shows that allocating clusters of tasks to robots can avoid some local minima that single item contracts become stuck in; although, C-contracts can get stuck in different local minima [17]. In early work on multi-robot auctions, Dias and Stentz developed a clustering

algorithm that connects geographically close tasks under the assumption that two tasks that are close have high inter-task synergies. Robots then exchange clusters of tasks through an auction method which experimentally is shown to perform better than single task auctions [3].

Many subsequent clustering approaches for MRTA problems use the distance between tasks as a metric for cluster formation. Sariel and Balch discuss the allocation of clusters of tasks to robots where in some situations optimal MRTA solutions can be generated that are unable to be formed using single item auctions, however, in other situations clustering performs worse [18]. Zlot and Stentz use K -means clustering to form clusters of geographically close tasks. To determine an ideal cluster, the value of k is incremented from 2 to the total number of tasks with the value of k generating clusters with the largest relative improvement over the previous value being used [26].

In our previous work [6,8], we expanded upon SSI auctions to develop *sequential single-cluster auctions* (SSC auctions). In SSC auctions a clustering algorithm forms fixed clusters of geographically close tasks which robots subsequently bid on using an SSI-like auction technique. Auctioning clusters of tasks reduces the numbers of bids required and thus reduces the communication overhead. This work also demonstrates that repeatedly forming clusters with different task memberships allows robots to consider many combinations of inter-task synergies that are not considered during SSI auctions which only allocate tasks once.

3.3 Dynamic Task Insertion

Despite a large body of work on auction-based algorithms for MRTA problems, few papers have considered the effects of dynamically appearing tasks in the problem domain. While it can be argued that algorithms that continually change task allocations *could* handle dynamically inserted tasks, this has little experimental grounding. An important consideration in the handling of dynamic tasks is deciding how much of the existing task allocation to modify. This can range from an individual robot locally replanning its task execution plan, through to all robots running auctions for a global reallocation of all uncompleted tasks.

Previous work by Schoenig and Pagnucco has considered SSI auctions with dynamically inserted tasks and compared the costs of robots bidding only for the new task versus a full new auction of all uncompleted tasks [19]. Their results show, despite a large trade-off in computational time, a global reallocation of tasks produces lower team costs than local replanning. Zlot *et al.* consider MRTA problems in an exploration domain in which a robot generates additional tasks for allocation after each task completion. These tasks are sequentially offered for auction to other robots, however, if no buyer is found the generating robot retains the task [27]. Viguria, Maxa and Ollero’s approach of repeatedly auctioning subsets of uncompleted tasks allows it to handle dynamically inserted tasks [23]. This approach sits between local replanning and global reallocation in that robots only offer for auction tasks that they specifically consider to be of high cost. Additionally, these approaches avoid the problem of never completing any

tasks through ensuring that the currently executing task is never offered for reallocation.

3.4 Tasks with Pickup and Delivery

Little work has focused on distributed auctions for MRTA problems with pickup and delivery. While it can be argued that many existing techniques for single point locations should continue to work for tasks with pickup and delivery, this has almost no experimental grounding. Despite this, a large body of work exists in the field of transport logistics.

Fischer, Müller and Pischel apply the CNP to transportation scheduling with fixed time windows [5]. In this work trucks bid for tasks from a central controller and can also make one-for-one swaps with other trucks before they begin to execute their plans. During the execution of plans, the trucks may face traffic delays and as such they can locally replan their routes or auction their uncompleted tasks. Their results show that global reallocation of uncompleted tasks provides a large reduction in distance travelled.

Kohout and Erol argue that Fischer, Müller and Pischel's generation of an initial allocation is poor and therefore global reallocation will produce much better results than local replanning [12]. In their analysis they study problems where multiple items can be transported together and additional jobs are announced sequentially. When a new job is announced each vehicle bids for the job according to the cost of completing the additional job relative to their existing commitments. To avoid problems where inserting additional tasks has large impacts on the completion time of other tasks, upon each task insertion, already scheduled tasks are permitted to be reallocated to other vehicles. In their empirical analysis they compare this approach to a popular operations research based approach [21]. Overall they show that their distributed approach is statistically equivalent to this centralised technique.

In a similar vein, Mes, van der Heijden and van Harten compare distributed auctions in MAS to hierarchical operations research approaches in dynamic environments [14]. In this work tasks arrive sequentially and trucks can only carry one task at a time. Each truck bid calculation for a task considers the time required to do the job and any waiting time between jobs before and after. During execution trucks can also swap future task commitments between each other to improve the overall solution. In the comparison to the operations research approaches the distributed auction approach performs substantially better in highly dynamic environments.

4 Repeated SSC Auctions with Dynamic Tasks

We now formally explain SSC auctions and provide a simple extension to handle dynamically appearing tasks. SSC auctions assign fixed clusters of tasks to robots over multiple bidding rounds. In Fig. 2 an algorithm describing this process is given. During the bidding stage (Lines 2-7) the robot calculates bids for every

function SSC-Auction (U, K_{r_i}, r_i, R)
Input: U : the set of clusters to be assigned
 K_{r_i} : the set of clusters presently assigned to robot r_i
 r_i : the robot
 R : the set of robots
Output: K_{r_i} : the set of clusters assigned to the robot

```

1: while ( $U \neq \emptyset$ )
2:   /* Bidding Stage */
3:    $\beta_{min} \leftarrow \infty$ 
4:   for each cluster  $C \in U$ 
5:      $\beta_{r_i}^C \leftarrow \text{CalcBid}(K_{r_i}, C)$ ;
6:      $\beta_{min} \leftarrow \min(\beta_{min}, \beta_{r_i}^C)$ ;
7:    $\text{Send}(\beta_{min}, R) \mid B \leftarrow \bigcup_i \text{Receive}(\beta_{r_i}^C, R)$ ;
8:   /* Winner-Determination Stage */
9:    $(r', C) \leftarrow \arg \min_{(r' \in R, C \in U)} B$ ;
10:  if  $r_i = r'$  then
11:     $K_{r_i} \leftarrow K_{r_i} \cup C$ ;
12:   $U \leftarrow U \setminus C$ ;

```

Fig. 2. Algorithm for Sequential Single-Cluster Auctions [8]

unassigned task cluster and submits its single lowest bid for any one cluster to all other robots. Each bid is a triple $\beta = \langle r_i, C_j, b_\lambda \rangle$ of a robot r_i , a task cluster C_j and a bid cost b_λ . Each bid calculation requires robots to provide a solution to the travelling salesperson problem taking into account the tasks they already have allocated and the tasks in the cluster for which they are calculating a bid. Because this problem is NP-hard, robots may use the cheapest-insertion heuristic to provide a close approximation to the optimal solution. At the conclusion of each bidding round (Lines 8-12), one previously unassigned task cluster $C = \{t_1, \dots, t_o\}$ is assigned to the robot that bids the least for it so that the overall team cost increases the least. After all task clusters $K = \{C_1, \dots, C_k\}$ are allocated, each robot seeks to minimise the distance travelled to complete all its allocated tasks. To achieve this, robots do not have to do all tasks in a cluster sequentially. When a robot is awarded a new cluster, the robot adds the tasks in this new cluster to its existing task assignment and replans its path to travel.

Before the SSC auction algorithm begins each task is assigned to one, and only one cluster, and clusters can be of varying sizes. All robots are informed of all tasks and all clusters before calculating bids. For the initial task allocation, either a centralised task manager or a single robot generates task clusters and subsequently informs all robots of the details of each cluster. During repeated auctions each robot individually forms clusters of its uncompleted tasks. After removing any clusters containing tasks that are currently being executed, each robot informs all other robots of their available clusters for auctioning (a formalisation of this algorithm is given in [8]). While the auction for uncompleted

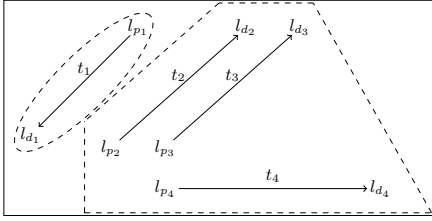


Fig. 3. Four tasks clustered into two pickup task clusters

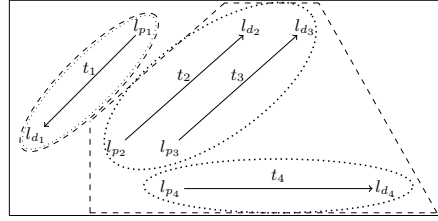


Fig. 4. Formation of three final task clusters based on delivery locations

task clusters runs, all robots in parallel, continue to complete their currently executing tasks.

To handle a new task t_n dynamically inserted into our system we must ensure that the valid complete solution to the task allocation problem $\cup_{r_i \in R} T_{r_i} = T$ continues to hold for $\cup_{r_i \in R} T \cup \{t_n\}_{r_i} = T \cup \{t_n\}$. The simplest way to meet this requirement is, upon the dynamic insertion of a new task, instantaneously assigning it to a robot $r_i \in R$. Depending upon the operating environment configuration, after the assignment the robot can either locally replan its path or can signal a repeated auction to globally reallocate and replan tasks across all robots. In our experiments a new task may be inserted immediately after a task delivery and is initially allocated to the robot that completed the delivery.

The formation of clusters of tasks that have both pickup and delivery locations is much more difficult than cluster formation in single task location problems. To generate low cost solutions to MRTA problems with pickup and delivery we need a clustering algorithm that considers the structure of tasks in the problem. In the development of our clustering approach, we considered three different approaches for generating a metric that describes each task. First, we attempted to cluster based on the midpoint of the line segment between the pickup and delivery location. However, this metric lacks any information about where either end point is located. Second, we attempted to cluster by forming vectors of pickup and delivery locations and chaining tasks that have a delivery location close to the pickup location of another task. This approach to clustering closely matches the bidding pattern of robots in auctions, and as a result, there would be no advantage over doing no clustering at all.

Our third, and successful, approach was to cluster the pickup and delivery locations of all tasks separately. First, we form clusters of tasks based on pickup locations (Fig. 3). Second, within each pickup location cluster we cluster again into smaller clusters based on delivery locations (Fig. 4). The complete set of all small clusters is then used by robots in bidding. We formulate this algorithm in Fig. 5. We begin with no final clusters set (Line 1). We initially cluster all tasks based on pickup location (Line 3). We then iterate over each of these pickup location clusters (Line 5). Within each pickup location cluster we cluster again

```

function TwoStepClustering ( $T, k_p, k_d$ )
Input:  $T$ : the set of tasks to be clustered
          $k_p$ : the number of pickup clusters to be formed
          $k_d$ : the number of delivery clusters to be formed
              per pickup cluster
Output:  $K$ : the set of clusters for auction

1:  $K = \emptyset$ ;
2: /* Pickup location clustering */
3:  $PickupClusters \leftarrow \text{CalcPickupClusters}(T, k_p)$ ;
4: /* Delivery location clustering */
5: for each pickup cluster  $p \in PickupClusters$ 
6:    $DeliveryClusters \leftarrow \text{CalcDeliveryClusters}(p, k_d)$ ;
7:    $K \leftarrow K \cup \{DeliveryClusters\}$ ;

```

Fig. 5. Clustering of tasks with pickup and delivery

based on delivery location (Line 6). Finally we merge each set of delivery location clusters into the final set of clusters for auction (Line 7).

Both clustering functions `CalcPickupClusters` and `CalcDeliveryClusters` consider only one side of a task's location. This allows us to use any existing clustering algorithm that clusters based on point locations, e.g., K -means clustering or single-linkage clustering. In our algorithm we need two k values k_p and k_d ; one for each clustering function. For our experiments we seek to find k overall clusters for auction such that $k_p * k_d = k$. This ensures that each pickup cluster is split into equal numbers of delivery clusters. However a problem can arise, due to each cluster containing a varying number of tasks, when considering the clustering of delivery locations inside a pickup cluster. If the number of tasks in the cluster $|p|$ is less than k_d we can only form $|p|$ clusters, and as a result, in total we will have less than k clusters. Without modifying the behaviour of the algorithm used to form the pickup clusters we cannot prevent this occurring. As a result, in these situations we end up with fewer clusters than we initially sought. To mitigate the effect of this during the formation of a large number of clusters we suggest a novel solution to gradually increase the value of k_d used in remaining cluster formations:

$$k_d = k_d + \frac{k_d - |p|}{|\text{remaining pickup clusters}|} \quad (5)$$

First we calculate the difference between the requested number of clusters k_d and the number of tasks in the cluster $|p|$. We then divide this by the number of remaining pickup clusters that have yet to have delivery clusters formed inside them. We add this value to k_d and continue to the next pickup cluster for clustering.

5 Empirical Experiments

We test SSC auctions with dynamic tasks requiring pickup and delivery in a simulated office-like environment with 16 rooms. Each room contains four doors that can be independently opened or closed to allow or restrict travel between rooms (Fig. 1). This environment has become the standard testbed in recent literature [11,25,16,6,8]. Each robot is supplied a map of the environment at a resolution of 510x510 grid units. Each grid unit is representative of a 5cm by 5cm area of space and gives an overall simulated space of 25.5m by 25.5m. In each experiment, the doors between different rooms and the hallway are either open or closed.

We test on 25 randomly generated configurations of opened and closed doors with each robot starting in a different random location. Robots can only travel between rooms through open doors and they cannot open or close doors. However, it is guaranteed there is at least one path between each room and every other room. For each configuration we test with 10 robots and 60 tasks. We use single-linkage clustering with a true path distance metric (previously discussed in [7]) for our clustering algorithm. For the initial allocation we form $k = \frac{1}{2}|T_{\text{known}}|$ clusters and where $T_{\text{known}} \in T$ is the set of known tasks at the start of the initial allocation. For repeated auctions each robot individually forms $k = \frac{1}{2}|T_{r_i}|$ clusters.

In each experiment configuration a robot may be randomly assigned a new task upon completion of a task delivery, we then compare local replanning versus global reallocation. In local replanning, when a robot is assigned a new task, the robot replans its path to complete all uncompleted tasks. In global reallocation, when a robot is assigned a new task it signals to all other robots to begin an auction of all uncompleted tasks across all robots. We compare three ratios of dynamic to static tasks, 25%, 50%, and 75% unknown at the start. We also compare our results to a baseline one-off task allocation with all tasks known. Finally, we compare the effects of different robot task capacities of 1, 3, and 5.

The mean results for the MiniMax team objective are presented in Table 1 and for the MiniSum team objective in Table 2. In considering capacity constraints, unsurprisingly, the absence of constraints produces the lowest team costs and the more restrictive the constraints the higher the cost. This directly leads to the largest reduction in team costs occurring in scenarios with highly restrictive constraints. In both team objectives, when robots are restricted to a capacity of only executing one task at a time, the global reallocation of tasks produces better results than the baseline of all tasks known. While this may initially come as a surprise it has been experimentally shown before in SSC auctions with static tasks [8] and SSI auctions with dynamic tasks [19]. The key explanation for this is that during a one-off task allocation, due to the greedy nature of SSI auctions, each robot can reach a local minima in its bidding preferences whereas in repeated auctions this is avoided. Overall, across both team objectives, global reallocation generally produced lower overall results than local replanning.

Table 1. Mean MiniMax Team Objective Results (percentage improvement of reallocation compared to replanning in brackets)

		Local Replanning		
Capacity	All Known	25% Unknown	50% Unknown	75% Unknown
1	7857	8228	7276	7275
3	3985	5430	4976	6256
5	3070	4620	4471	6250
∞	2602	4438	4418	6250
		Global Reallocation		
Capacity	All Known	25% Unknown	50% Unknown	75% Unknown
1	7857	5228 (36.5%)	5593 (23.1%)	5911 (18.7%)
3	3985	3800 (30.0%)	4389 (11.8%)	4877 (22.0%)
5	3070	3469 (24.9%)	3800 (15.0%)	5000 (20.0%)
∞	2602	3415 (23.1%)	3810 (13.8%)	5165 (17.4%)

Table 2. Mean MiniSum Team Objective Results

		Local Replanning		
Capacity	All Known	25% Unknown	50% Unknown	75% Unknown
1	41539	41527	42025	45305
3	21245	26922	28316	33238
5	16519	22554	25720	30096
∞	10150	16613	19675	27985
		Global Reallocation		
Capacity	All Known	25% Unknown	50% Unknown	75% Unknown
1	41539	37449 (9.8%)	40278 (4.2%)	36907 (18.5%)
3	21245	25775 (4.3%)	28303 (0.0%)	26594 (20.0%)
5	16519	22639 (-0.4%)	24692 (4.0%)	22238 (26.1%)
∞	10150	17813 (-7.2%)	22207 (-13%)	20332 (27.3%)

Across the MiniMax team objective results, the average advantage of global reallocation over local replanning ranges from 11.8% to 36.5%. For local replanning the best results occur when 50% of tasks were unknown. We speculate that in situations where 25% of tasks are unknown, despite being “better informed” of other tasks during planning, there may be instances where new tasks are inserted late into plan execution which cause robots to travel greater distances. This hypothesis is further supported by the large improvement gains shown by global reallocation in the 25% unknown experiments. Across the global replanning results the best results occur when only 25% of tasks are unknown. We speculate that in these instances the advantage of being “better informed” of other tasks helps with the formation of new clusters and repeated auctions of tasks. Overall, in the worse case of 75% tasks unknown, on average, a robot travels a maximum of twice the distance of the baseline result.

The MiniSum team objective results show a much smaller benefit in global reallocation over local replanning. These differences range from a 13% increase

Table 3. Mean Initial Task Allocation Computation Time (s) For MiniMax Team Objective

Capacity	All Known	25% Unknown	50% Unknown	75% Unknown
1	268	145	62	16
3	265	145	61	16
5	264	144	61	16
∞	242	137	60	16

Table 4. Mean Overall Cumulative Task Allocation Computation Time (s) For MiniMax Team Objective

		Local Replanning		
Capacity	All Known	25% Unknown	50% Unknown	75% Unknown
1	268	173	90	41
3	265	172	89	42
5	264	171	88	42
∞	242	157	85	41
		Global Reallocation		
Capacity	All Known	25% Unknown	50% Unknown	75% Unknown
1	268	347	276	119
3	265	245	175	106
5	264	226	176	105
∞	242	206	174	105

to a 27% decrease in total distance travelled. Local replanning produces the best results when only 25% of tasks are unknown at the start. Contrarily in global reallocation, the largest improvements overall local replanning come in the highly dynamic environment of 75% of tasks being dynamically inserted. We speculate that the reason for this is due to the high numbers of repeated cluster formations exposing many different inter-task synergies during each repeated auction. Additionally, in scenarios with no capacity constraints local replanning outperforms global reallocation. Again, in the worst-case scenario the maximum distance result was twice that of the corresponding baseline result.

In addition to measuring the distances robots travel, we also consider the amount of computational real-time robots require to generate an initial allocation of tasks, and then the accumulated time required to repeatedly replan or reallocate tasks. The experimental timing results are from a system with a 2.8GHz Intel Core i7 CPU, 8GB RAM, running Ubuntu 11.04 x64.

Table 3 shows the mean time required to cluster and auction an initial allocation of tasks for the MiniMax team objective. The results for the MiniSum team objective are not shown, however, they are near identical. Unsurprisingly, the fewer the number of tasks known, the faster the initial allocation of tasks. In the most dynamic situation the generation of an initial allocation is 15 times quicker than the baseline. This is an important result because the quicker an initial allocation is generated, the sooner robots can begin executing tasks.

We also note that the task capacity constraint has minimal influence upon the time required to generate the initial allocations.

The overall cumulative time required for robots to replan or reallocate upon the insertion of all dynamic tasks into the system is given in Table 4. In the worst-case, global reallocation of tasks is three times slower than local replanning. Of particular interest is that in all dynamic situations, local replanning was substantially quicker in computational time than the baseline. Also in highly dynamic situations global reallocation generated solutions faster than the baseline.

Overall, taking into consideration mean distance and computational time results we can conclude that when robots seek to achieve a MiniMax team objective it is best for robots to work together and globally reallocate tasks. However, when robots seek to achieve a MiniSum team objective, except in highly dynamic environments, the small improvement offered by global reallocation is offset by much higher computational times and in many situations would be of little benefit.

6 Conclusions and Future Work

In this paper we have built upon previous work on SSC auctions and demonstrated their effectiveness in task allocation with pickup and delivery. Our empirical analysis considered the trade-off in performance between local replanning and global reallocation for dynamic task allocation. Our key result shows that global reallocation generally produces lower team costs than local replanning. However, to achieve this there is a large computational time cost.

In the consideration of future work, an ongoing challenge in cluster formation is determining suitable values for k . There are a number of more complex clustering algorithms that do not require a pre-set number of clusters to form, such as DBSCAN [4], and their usefulness in SSC auctions could be considered. Another aspect of clustering to consider is the number of items in each cluster. At present we impose no limit on the number of items, or any preference to allocating large clusters first. If a fixed maximum limit on the number of items in each cluster was imposed it could produce more even clusters which may affect the task allocation results.

References

1. Berhault, M., Huang, H., Keskinocak, P., Koenig, S., Elmaghraby, W., Griffin, P.M., Kleywegt, A.J.: Robot exploration with combinatorial auctions. In: IROS, pp. 1957–1962 (2003)
2. Dias, M.B., Zlot, R., Kalra, N., Stentz, A.: Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE* 94(7), 1257–1270 (2006)
3. Dias, M.B., Stentz, A.: Opportunistic optimization for market-based multirobot control. In: *Intelligent Robots and Systems*, vol. 3, pp. 2714–2720. IEEE (2002)
4. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: *KDD*, pp. 226–231 (1996)
5. Fischer, K.: Cooperative transportation scheduling: An application domain for *dai*. *Applied Artificial Intelligence* 10(1), 1–34 (1996)

6. Heap, B., Pagnucco, M.: Sequential single-cluster auctions for robot task allocation. In: Wang, D., Reynolds, M. (eds.) AI 2011. LNCS, vol. 7106, pp. 412–421. Springer, Heidelberg (2011)
7. Heap, B., Pagnucco, M.: Analysis of cluster formation techniques for multi-robot task allocation using sequential single-cluster auctions. In: Thielscher, M., Zhang, D. (eds.) AI 2012. LNCS, vol. 7691, pp. 839–850. Springer, Heidelberg (2012)
8. Heap, B., Pagnucco, M.: Repeated sequential auctions with dynamic task clusters. In: AAAI (2012)
9. Koenig, S., Keskinocak, P., Tovey, C.A.: Progress on agent coordination with cooperative auctions. In: AAAI (2010)
10. Koenig, S., Tovey, C.A., Lagoudakis, M.G., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A.J., Meyerson, A., Jain, S.: The power of sequential single-item auctions for agent coordination. In: AAAI, pp. 1625–1629 (2006)
11. Koenig, S., Tovey, C.A., Zheng, X., Sungur, I.: Sequential bundle-bid single-sale auction algorithms for decentralized control. In: IJCAI, pp. 1359–1365 (2007)
12. Kohout, R.C., Erol, K.: In-time agent-based vehicle routing with a stochastic improvement heuristic. In: AAAI, pp. 864–869 (1999)
13. Lagoudakis, M.G., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A.J., Koenig, S., Tovey, C.A., Meyerson, A., Jain, S.: Auction-based multi-robot routing. In: Robotics: Science and Systems, pp. 343–350 (2005)
14. Mes, M., van der Heijden, M., van Harten, A.: Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems. *European Journal of Operational Research* 181(1), 59–75 (2007)
15. Nanjanath, M., Gini, M.L.: Dynamic task allocation for robots via auctions. In: ICRA, pp. 2781–2786 (2006)
16. Nanjanath, M., Gini, M.L.: Repeated auctions for robust task execution by a robot team. *Robotics and Autonomous Systems* 58(7), 900–909 (2010)
17. Sandholm, T.: Contract types for satisficing task allocation: I theoretical results. In: Proceedings of the AAAI Spring Symposium: Satisficing Models, pp. 68–75 (1998)
18. Sariel, S., Balch, T.R.: Efficient bids on task allocation for multi-robot exploration. In: FLAIRS Conference, pp. 116–121 (2006)
19. Schoenig, A., Pagnucco, M.: Evaluating sequential single-item auctions for dynamic task allocation. In: Li, J. (ed.) AI 2010. LNCS, vol. 6464, pp. 506–515. Springer, Heidelberg (2010)
20. Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Computers* 29(12), 1104–1113 (1980)
21. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research* 35(2), 254–265 (1987)
22. Tovey, C., Lagoudakis, M., Jain, S., Koenig, S.: The generation of bidding rules for auction-based robot coordination. *Multi-Robot Systems III*, 3–14 (2005)
23. Viguria, A., Maza, I., Ollero, A.: Set: An algorithm for distributed multirobot task allocation with dynamic negotiation based on task subsets. In: ICRA, pp. 3339–3344 (2007)
24. Zhang, K., Collins Jr., E.G., Shi, D.: Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction. *TAAAS* 7(2), 21 (2012)
25. Zheng, X., Koenig, S.: K-swaps: Cooperative negotiation for solving task-allocation problems. In: IJCAI, pp. 373–379 (2009)
26. Zlot, R., Stentz, A.: Complex task allocation for multiple robots. In: ICRA, pp. 1515–1522 (2005)
27. Zlot, R., Stentz, A., Dias, M.B., Thayer, S.: Multi-robot exploration controlled by a market economy. In: ICRA, pp. 3016–3023 (2002)

AMASON: Abstract Meta-model for Agent-Based Simulation

Franziska Klügl¹ and Paul Davidsson²

¹ School of Science and Technology
Örebro University, Örebro, Sweden
franziska.klugl@oru.se

² School of Technology
Malmö University, Malmö, Sweden
paul.davidsson@mah.se

Abstract. The basic prerequisite for methodological advance in Multi-Agent Based Modelling and Simulation is a clear, ideally formally-grounded, concept of our subject. A commonly accepted, implementation-independent meta-model may improve the status of MABS as a scientific field providing a solid foundation that can be used for describing, comparing, analysing, and understanding MABS models. In this contribution, we present an attempt formalizing a general view of MABS models by defining the AMASON meta-model that captures the basic structure and dynamics of a MABS model.

1 Introduction

In this contribution, we present AMASON (Abstract Meta-model for Agent-based Simulation) as a general formalization of Multi-Agent Based Simulation (MABS) models. As the example of Agent-Oriented Software Engineering (AOSE) shows, a well-defined meta-model provides a solid foundation for all processes relevant for “engineering” MABS models: specification and documentation, appropriate tool support, teaching and exchange and reuse of models. A meta-model provides a clear set of categories and a language for describing a MABS model. It can be used in combination with documentation frameworks such as ODD [14]. ODD gives a structure for documentation, yet no clear language. Describing different MABS models using concepts taken from the same meta-model supports their comparison and comparative analysis, especially if the models are described without programming language level details. This argument that can be also found in [20] suggesting the use of explicit model ontologies. A prerequisite is hereby that the meta-model is formulated in an implementation- and platform-independent way.

Last but not least, teaching MABS development is facilitated if a concise meta-model is available, as fuzziness of terms and structures are avoided. MABS could be handled and introduced in the same rigorous way as other microscopic simulation paradigms with a clear definition of their basic structures. Practical introductions such as [27] could be augmented with a theoretical introduction stating which elements a MABS model has and how they relate to each other. Clearly describing the scope of MABS

and also clearly stating what MABS models are not, is supported by an appropriate meta-model.

Thus, the existence of a shared meta-model would improve the status of MABS as a scientific and engineering field. Within the scope of this contribution with its limited number of pages, we can just sketch the first considerations towards such a meta-model, called AMASON. Many aspects that are especially important for the social sciences, such as negotiations, norms and organization, are left to future work, as we first wanted to find an abstract and general common ground that also includes domains beyond social science simulation. Moreover, we do not want to define the meta-model on a too detailed level including e.g. data types, but instead throw light on the really important aspects. Nevertheless, a generic, abstract approach leaves space for more specific extensions and specializations.

In the remainder of the paper we will first give the background for our proposal, discussing mainly related work from AOSE and MABS. In section 3, we introduce the basic structural and dynamic elements of the meta-model in its current status. We end by wrapping up and discussing future work. Due to space limitation, we cannot give a consistent full example using the meta-model. Yet, the description of the meta-model is illustrated with the basic Sugarscape model [6] as running example.

2 Background and Related Work

2.1 Meta-models in Multi-agent Software

Introductory literature on multi-agent systems such as [32] uses formal descriptions of what is an agent, how it is embedded in an environment and indicate abstract architectures. Although apt as a basic introduction to agents (in form of a one-agent system), the value of these descriptions for capturing the idea of multi-agent systems is limited. A more elaborated formulation of meta-models for multi-agent systems can be found in [5] and its successors using *Z* schemata starting from the definition of an “object” refining it to “Agent” and “Autonomous Agent”. Their meta-model is very detailed also including statements on goals and motivations, commitments and obligations and relations between agents in a multi-agent system with respect to collaboration.

Concisely defined meta-models for multi-agent systems play an important role in the area of AOSE. A meta-model hereby defines the framework of concepts and their relations providing a language for analysis and specification. Their clear and formal definition received a lot of attention especially together with topics such as model-driven design or method fragmentation. Only, if the meta-model is absolutely clear, interfaces between models on different abstraction levels or for different aspects can be created or combined. Thus, it is not surprising that there is a wealth of meta-models used in AOSE. UML-based specifications of meta-models for ADELFE, Gaia and PASSI can be found in [2]. The AOSE methodology INGENIAS and its meta-model have been successfully applied to agent-based social simulation [9].

Hahn et al. [15] introduce a platform-independent meta-model integrating different views from an multi-agent view to an environmental view. Also, meta-models have

been defined specially for the specification of organizational concepts; see [7] for the ALAADIN meta-model providing the terms agent, group and role as core organizational concepts. Also Beydoun et al. [3] aim at providing a general meta-model and hence a modelling language that can be used for describing agents independent from an application.

Yet in all those AOSE meta-models focussing on organizational concepts, (virtual) time and spatial environment play none or just a subordinate role, but would be central in simulation settings. A model that clearly shows the relevance of the part of the overall system representing the environment is the “influences and reactions” model [8]. It gives a precise understanding of the relation between the agents’ actions with the agent want to influence its environment and the reaction of the environment that finally determines what the agents’ action is actually affecting. Recently, it has been further developed into a general concept of interaction in the MASQ meta-model [4]. In addition to the Influence-Reaction principle, R. Dinu et al. hereby postulate the separation of mind and body, as well as a segmentation of space (and four more principles including culture). Currently, MASQ is more a family of meta-models avoiding particular design choices.

Another meta-model with a strong focus on environmental entities is the Agents& Artifacts model [25] with a clear distinction between agents and “reactive objects” that – as explicit environmental entities – provide services and thus support the interaction between the actual agents. Thus, the A&A meta-model forms an abstraction from the real environment towards a framework that supports the design and implementation of Multi-Agent Systems.

There are several reasons why the AOSE meta-models are not appropriate for describing MABS models. A fundamental reason is the fact that a MABS model is a virtual representation of another system (including individuals, objects, etc.), whereas a MAS is an artifact that interacts with an environment. The meta-models on which the different AOSE methodologies are based, nevertheless can be valuable as they provide concepts for analysing, describing and specifying multi-agent models. Yet, a modeller must be aware about the assumptions that the usage of those meta-models created for supporting the design of a Multi-Agent System impose on the final model. Meta-models which are based on grounded social science theories of institutions or similar would take that burden from the modeller.

2.2 Conceptualizations in MABS

There is clearly a lack of formal specification in MABS. Introductory textbooks such as [13], [23] or [27] are based on heuristics and best practices. They do not give a formal and precise grounding beyond a textual characterization what a MABS model is and may contain. Similarly, the ODD Protocol [14] provides a framework for documentation that clearly advices which elements are necessary for a full documentation. However, ODD is not giving a meta-model or language that a modeller can use for capturing the elements of the MABS in the different submodels. Bandini et al. [1] give a good conceptualization of elements of multi-agent systems relevant for MABS. Yet, without formalization their approach remains nevertheless fuzzy in detail.

During the last years, a number of approaches have been published with a similar objective as in this contribution. An early meta-model can be found in [18]. There, an explicit separation between resources and agents is made which is distinct from our more transparent characterization of bodies and minds. Another important difference is the environmental structure that has not been elaborated earlier.

The reference model proposed in [28] is fully based on an event-based approach, in which sensor and effector activation create cascades of events that may manipulate the internal state of the agent as well as its external environment. Constraints are used for representing events that do not work in a particular environmental configuration. One aspect that [28] explicitly points out is that actions – which are conceived as activation of effectors – take time. The only atomic element in the reference model is the event.

There are several formal frameworks for representing simulation models starting from the system science-inspired DEVS framework of Object-Oriented Simulation [33]. Approaches such as AgedDEVS [29], form the formal model underlying the JAMES II system. DEVS/RAP [17] or PECS [30] reference model can be seen as extensions of Object-Oriented Simulation models integrating more or less sophisticated agent structures for capturing internal agent processes, belief or plan structures as well as variable overall system structures with agents that are generated, die or change interaction partners. Mueller [22] systematically analyses which extensions of the DEVS meta-model for event-based Object-Oriented Simulation [33] capturing parallel simulation and variable structures, can best serve as a basis for MABS models. An advantage of DEVS is the precisely defined semantics of basic elements and different update functions. Due to its generic nature based on the notion of “system” with input, internal state and output, it can be seen as forming the starting point for many, if not all meta-models in agent-based simulation.

A generic formalization of MABS concepts has been suggested by Helleboogh et al. [16]. They focus on interactions between agents’ activities and an explicit environment. Dynamism is elaborated based on the concept of activities. Activities represent the agents’ influences, “reaction laws” determine how the environment reacts on those and transforms them. Interaction is handled using “interference laws”. The overall goal is to clearly define environmental dynamics. Also [31] use the influences and reactions idea of [8] as a starting point, but focus on non-global synchronization as it is necessary for truly distributed applications.

Recently, two suggestions for MABS meta-models have been published that can be seen as attempts to develop meta-models similar to the AOSE meta-models, yet adapted to challenges of (social science) MABS simulation: MAIA [12] and easyABM [10] provide languages with a focus on the societal level. MAIA formalizes social institutional theories, easyABM provide an overall detailed view that is apt for code generation yet shares many problems with the traditional AOSE approaches sketched above.

Thus, there is currently no meta-model that provides a basic view on what a MABS model is and contains, expressed using a minimal set of concepts that is applicable to all types of models – ranging from social science to models with simpler agents.

3 A Generic Meta-model: AMASON

AMASON, the meta-model that we propose originates from our collected experience of building and analysing MABS models in various domains. As we aim at a simple model that makes the core concepts explicit, we cannot take a meta-model of for example object-oriented simulation and add concepts. Also, existing meta-models for MABS platforms, such as [26] or [24] are far too detailed and specific containing low level data types.

AMASON contains two basic views: the context of the model and the contents of the model. In the following we focus on the model itself. The context of the model is then needed for setting up a simulation based on the model. It is essential for developing, validating and using the model. The model context contains information about the objectives, how the model shall be used and under with which parameter configurations it can be used, information on how and with what data the model is validated, etc. Whereas these aspects are relatively obvious considering standard simulation literature, conceptual confusion occurs when one attempts to come up with a underlying conceptualization of the actual MABS model. Despite of their importance for documenting the model, in this contribution we concentrate on the core content of the meta-model as this can be used for formulating a more structured context model.

3.1 Elements of a Model

We identified three types of components for a multi-agent simulation model: *Body*, *Mind* and *Regions*. All three may possess some form of internal state. The idea behind is to separate the physical entity from the mental one to make embodiment explicit and also provide a clear distinction between entities that possess reasoning capabilities and entities that do not, but nevertheless populate the environment and are of use (or obstacles) for the active entities capable of decision making. The artefact framework [25] shows that this conceptual separation is also relevant for multi-agent system software. The idea behind *regions* is to provide a uniform perspective on the large variety of spatial environments. Whereas a body is located in a region, a mind needs to be connected to a body for achieving situatedness. An agent consists of both *body* and *mind*.

Body. A *body* represents a physical entity in a MABS model: A human or an animal body, the physical parts of a robot, but also rocks, food items or houses. The *body* forms the “hardware” of an agent. It carries sensors and actuators and thus provides the technical means to interact with the environment. Thus, it is the *body* which is located on a specific *region*.

Each *body* has a state that is domain- or model-specific. The state may be structured into a set of parameters and state variables that may contain arbitrary complex data ranging from numbers denoting energy storage to structure representing a complex metabolism. States do not need to be discrete, yet the *body* is in a particular state at a particular time in a particular simulation run.

The state of a *body* can be updated by *region*-specific processes, which are captured by the state of the *region*. For example the temperature of a *body* may rise if the

temperature of the **region** is high. Such state changes may happen without any decision making of a **mind**. In a more formal treatment: the set of all **body**-elements in a simulation model is $B = \{b_1, b_2, \dots, b_n\}$ with b_i as an individual **body** with index i . Every **body** has an individual set of possible internal states described by Σ_{b_i} . The state of **body** b_i at time t is denoted with $s_{b_i}(t) \in \Sigma_{b_i}$. The initial state is $s_{b_i}(t_0) \in \Sigma_{b_i}$ where t_0 is the point in time when b_i is generated.

Mind. To become an (intelligent) agent, a **body** needs to be coupled to a **mind**. This entity contains reasoning capabilities, thus handles the decision-making processes. It possesses also an internal memory or state whose structure and content is depending on the particular reasoning mechanism. As in the **body**, we do not restrict the structure or representation languages used for expressing a **mind** state. A typical structure might be a BDI architecture, the state would then contain current beliefs, goals, and committed plans. In a different model instantiation, a **mind** may be based on a neural network - the weights of the neural network then would correspond to the current state of the **mind**.

We denote the set of all **minds** with the letter $M = \{m_1, m_2, \dots, m_k\}$ with m_j as an individual **mind** with index j . The number of **minds** $k \leq n$ with n as the number of **bodys**. Every **mind** may have an individual set of possible internal states Σ_{m_j} being the set of possible states of **mind** with index j . The state of **mind** m_j at time t is denoted with $s_{m_j}(t) \in \Sigma_{m_j}$. The state hereby is depending on the particular reasoning approach that the **mind** m_j uses (see section 3.2). The initial state is $sm_{m_j}(t_0) \in \Sigma_{m_j}$ where m_j denotes the particular **mind**, t_0 is the point in time when m_j is generated.

We call the coupling between **body** and **mind** the “embody” relation: *embody* : $M \rightarrow B$ with M as the set of **minds** and B as the set of **bodys**. A **body** without a **mind** corresponds to a passive object in the environment.

Region. A central part of a MABS model is the spatial environment where the agents and objects are situated. A large variety of spatial representations is in use: models with discrete or continuous maps, cellular automata, network structures with or without metrics; there are also models without explicit space. In all cases (except those in which all agents are virtually at one location, which in [21] are called “aspatial soups”), an explicit environmental structure is necessary for supporting the representation of locality. We suggest the idea of connected **regions** as a general, yet structured way of conceiving a heterogeneous model of the spatial environment of a MABS model.

Every **region** has its specific spatial representation (continuous, discrete or aspatial). A **region** is conceived as an explicit entity with a state. Thus, possibly heterogeneous global properties, such as temperature or light can be captured in a way similar to a **body**.

Connections between the **regions** form the edges of a network of **regions**. They may be dynamic - that means they may be created by agents or destroyed. How a connection looks like, is depending on the particular spatial model used in the **regions**. For instance, a connection may be a (directed) link without structure or it may be a door with a given width. It is quite obvious how continuous or discrete maps can be conceived as one **region** or that a pure network corresponds to connected **regions**, where one **region**

is one aspatial node of the network. Yet, the concept also supports more complex spatial structures:

- Grid maps consist of cells that carry information which is accessible for the agents located on a cell. This information may be a discretized gradient field for a pedestrian simulation. Each cell would hereby correspond to a **region**, its state represents the gradient value. Every cell-**region** has connection to its neighbouring cells. The cell itself has no map, all agents on that cell have no further detailed position. A cellular automaton can be conceptualized by adding an agent to each cell which does not move, but intentionally updates the state of the cell.
- A second interesting case are metric networks: network structures in which the nodes have positions in some metric space and the length of connections between nodes are relevant for the agents populating such an environment. Examples are road networks. For expressing such an spatial structure, a network of connected **regions** needs to be located within on higher-level **region** containing a continuous or discrete map.

The consequence of these considerations is that the meta-model should allow for hierarchies of **regions** so that, e.g., a network of **regions** can be located on/in a **region**. This **regions** within **regions** concept together with the explicit connection model forms the appropriate meta-model for capturing all possible environmental representations that we encountered so far.

Capturing **regions** in a more formal language gives: As for **bodys** and **minds**, the set of all **regions** is $R = \{r_1, r_2, \dots, r_l\}$ with r_x as an individual **region** with index x . The state of **region** r_x at time t is denoted with $s_{r_x}(t) \in \Sigma_{r_x}$ – similar to the **body**.

Regions are connected with explicit connections. The structure of a connection is depending on the particular structure of the **region**: if the **region** is a container without internal spatial structure, the connection corresponds to a link; if the **region** has a 2-dimensional grid structure, a connection maps a set of grid cells on **region** r_x to a set of grid cells on **region** r_y . If the **region** has a 2-dimensional continuous map, a connection maps a set of positions (line) at the edge of one **region** to the other. In case of a 3-dimensional map, the connection may not just be a line, but could be also a part of plane. Also connections between different forms of **regions** are possible, such as connecting a 2-dimensional grid (e.g. representing some ground surface) to a 3-dimensional continuous map (e.g. representing the atmosphere). Thus, we define a “connection area” C which’s particular form depends on the **regions** that it connects. Connection are then represented by a function $connect : R \times R \rightarrow C$.

The *locate* function assigns a position in a **region** to a **body**: $locate_{r_x} : B \rightarrow P_{r_x}$. The *locate* function is depending on the particular **region**. P_{r_x} is the set of possible positions in that **region** r_x . What a position can be is depending on the **region**. For a continuous **region** with m dimensions, the set of possible positions would be $P_{r_x} = \mathbb{R}^m$.

Figure 1 gives an overall summary of the different elements of the AMASON meta-model.

Illustration: Sugarscape. We use the famous Sugarscape model for illustrating the elements of AMASON presented so far. In the Sugarscape world (we refer to the initial,

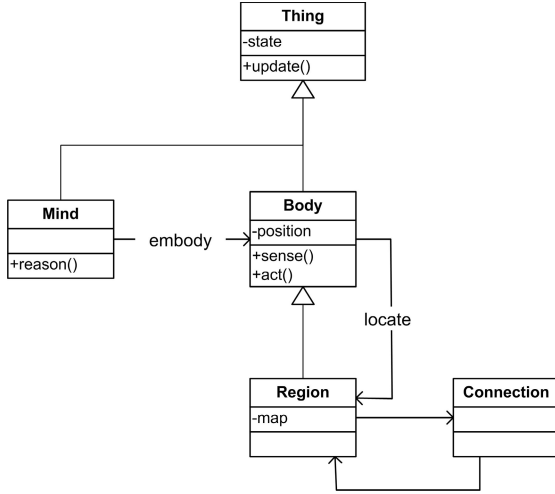


Fig. 1. Overview over the elements in AMASON

simple version that is described in chapter 2 of [6]), agents move on a grid harvesting and consuming “sugar”.

Following the concepts above, a **region** corresponds to a sugar cell. All **region**-cells possess of same structure, yet not the same state: The state of a region is expressed by a vector of one variable and two parameters: *CurrentSugar*, *GrowthRate* and *SugarCapacity*. As these are modelled as natural numbers, the set of possible states is: $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$. The position of each cell is determined implicitly by its connections to its four neighbours (as we assume a torus, all **regions** have four neighbours). Explicitly modelled, there are four connections: *C_{west}*, *C_{east}*, *C_{south}* and *C_{north}* for the respective neighbouring relation. An explicit grid position would help setting up the landscape. The initial state values for the variables are given by the two sugar hills.

Agents – consisting of a **body** and a **mind** – are located on a **region**. More precisely: The **body** of an agent is located on a **region**. There is just space for one **body** on a cell. Thus, the *locate* : $B \rightarrow R$ is assigning one **body** to one **region**. We assume that all agent parameters and variables are assigned to the **body**, representing the metabolism of sugar consumption. The **mind** is responsible for the decision making about where to go next. The state of a Sugarscape **Body** is captured by two constants and one variables: *PerceptionDistance* \times *Metabolism* \times *SugarStorage*. The constants are parameters with individual values for each agent, but an agent cannot change them. Assuming that perception is restricted to a max limit of 6 cells and that neither the consumption of sugar nor the potential storage of sugar is limited, the set of all possible states of a Sugarscape **body** Σ_{b_i} is: $\{1, 2, 3, 4, 5, 6\} \times \mathbb{N} \times \mathbb{N}$. The position of the **body** can be determined via the *locate*-function. Hereby, we leave open whether in praxis, the **body** stores the connection to the **region** or the **region** possesses a list of **bodys** on it. Due to the simplicity of this agent, its **mind** has an empty state. Later we will introduce a

variable with temporary values of cells for facilitating the agent dynamics. Real extension involving e.g. Trade would require parameters (thresholds when to sell) and variable information on the budget.

3.2 Dynamics

Based on the basic structural concepts presented above, the relevant processes are defined: For expressing dynamics, an explicit representation of time is necessary: T is the time domain. The nature of the time set influences how the model can be executed. A usual choice is $T = \mathbb{N}$ with state changes only at discrete points in time. Whether a model is time- or event-driven is regarded as an implementation issue. The AMASON meta-model should be independent from how a simulation is implemented.

Dynamics in a MABS model can be associated with the following elements: a) the environment dynamics that cannot be influenced by the agents (cf. exogenous processes and events), b) the agents that intentionally manipulate their environment and themselves.

Environmental Dynamics. Environmental dynamics happen without being triggered by an agent. Processes such as seasonal temperature dynamics, a tree growing, or rain starts to fall or a stone is heating up are examples. In the meta-model we associate such dynamics with **regions**. One can distinguish between dynamics that just affect the state of the **region**, and dynamics that affect the state of **bodys** that are located on the **region**:

- Let $s_{r_x}(t) \in \Sigma_{r_x}$ be the state of the region r_x at time t . Then, we can capture the purely **region**-related dynamics by: $updateRegion : \Sigma_{r_x} \times T \rightarrow \Sigma_{r_x}$. With this function the state of the **region** changes over time without external triggers. With this, seasonal dynamics can be expressed. With humidity as a variable of the **region**, also the effects of events such as rainfall can be captured.
- Let Σ_{r_x} be the set representing possible states of the region r_x and $B|_{r_x}$ are the **bodys** that are currently located on r_x . The function $updateBodyState : \Sigma_{r_x} \times \Sigma_{B|_{r_x}} \times T \rightarrow \Sigma_{r_x} \times \Sigma_{B|_{r_x}}$ updates the state of all **bodys** that are located on the **region** based on their previous state and the state of the **region**. This is the right level to formulate that the body of a stone heats depending on the temperature of the **region**, on which it is located. With mobile agents the domain of the function is changing depending on the **bodys** that are at each time on the **region**, so a precise, closed mathematical formulation may be difficult.

Agent-Based Dynamics. Following the general concept of an agent, there is no doubt that agent dynamics follow a *sense-reason-act* process combination. This must be integrated with the distinction in **body** and **mind**. As they belong to the physical part of an agent, sensors and effectors are associated with its **body** yet reasoning with the **mind**. As “interfaces” between sensing and reasoning, we conceive “perception” that translates the sensed data into information that is relevant for reasoning. The reasoning then produces actions. Depending on the **body** state actions are transformed into executable actions (commands of the effectors) which then are handed over to the **region**,

on which the **body** is located. The **region** is responsible for producing the actual effects of all actions of agents on it. More formally, agent-based dynamics involves processes at **bodys**, **minds** and **regions**:

- **Body**: $sense_{b_i} : \Sigma_{R|b_i} \times \Sigma_{b_i} \times T \rightarrow \Sigma_{b_i}$ with $R|b_i$ denotes a set of **regions** that are currently accessible for the sensors of b_i . This is a simplification replacing an elaborated concept of a local environment. Movement of the agent again would result in a varying definition of the *sense* function, as different **regions** might be relevant. The **region** on which the **body** b_i is located should be part of this set. The results of the sensing process may be stored in a sensor memory with particular states as well. For reasons of simplicity, we assume that this sensor memory is part of the physical state of the **body**.
 $act_{b_i} : Act_{m_j} \times \Sigma_{b_i} \times T \rightarrow ExecAct_{b_i}$ is the function that transforms actions Act_{m_j} as instructions from the **mind** m_j that is controlling the **body** b_i ($embody(m_j) = b_j$) to actions $ExecAct_{b_i}$ that the **body** can actually do depending on its current state.
- **Mind**: The **mind** m_j possesses three processes that mirror the classical *perceive-reason-act*: $perceive_{m_j} : \Sigma_{b_i} \times \Sigma_{m_j} \times T \rightarrow \Sigma_{m_j}$ with $embody(m_j) = b_j$, processes the sensor data stored in the state of the **body** to information relevant for the **mind**. With this information, the current state of the **mind** is also updated in $update_{m_j} : \Sigma_{m_j} \times T \rightarrow \Sigma_{m_j}$ expressing some internal reasoning. Then the **mind** selects an action: $action_{m_j} : \Sigma_{m_j} \times T \rightarrow Act_{m_j}$. With Act_{m_j} as the set of all possible actions that the **mind** m_j may come up.
- **Region**: When the effectors of the **body** b_j execute a given executable action, this does not automatically mean that the environment changes. The actual effect on the environment is handled by the **region** r_x , which takes all the executable actions of all **bodys** that are located on it, for determining the actual effect of the combined actions: $effect_{r_x} : ExecAct_{B|r_x} \times \Sigma_{r_x} \times T \rightarrow \Sigma_{r_x}$.

Illustration: Sugarscape Continued. Starting from the above given structure, the next step is to specify the different functions describing the dynamics: We start with the environment: The increase of the *CurrentSugar-Value* by the growth rate upto the capacity of the cell. *GrowthRate* and *SugarCapacity* are constants.

$$updateRegion(< currentSugar, growthRate, sugarCapacity >, t) = \\ < \min(currentSugar + growthRate, sugarCapacity), \dots >$$

This is followed by the regions-specific influence on **bodys**. This contains two elements: the consumption of sugar in the **body** metabolism and the harvesting of sugar. The *SugarStorage* of the **body** is reduced by the value of *metabolism*, but increased by the *currentSugar* of the cell on which the **body** is located. The value of the *currentSugar* variable of the cell is set to 0:

$$updateBodyState(< currentSugar, \dots >, < \dots, metabolism, currentSugar >, t) = \\ (< 0, \dots >, < \dots, metabolism, (currentSugar - metabolism) + r.currentSugar >)$$

If the *sugarStorage* of the **body** becomes negative, the **region** deletes the **body** together with its corresponding **mind**.

For deciding where to go next, the agent must first determine the subset of sugarcells that it can sense. The size of that set of cells forming the domain of the *sense*-function, depends on the value of *PerceptionDistance*. So with *PerceptionDistance* = 2, the domain of the *sense_b*-function is a set of cells with $c_b = \text{locate}(b): \{b, c_{west}(b), c_{west}(c_{west}(b)), c_{south}(b), \dots\}$. For storing the result of sensing, we introduce the before mentioned variable *b* containing a datastructure with a table of the current sugar values in the neighbourhood with the direction towards them: *SugarPercept*.

The *percept* function of the **mind** transfers the information in the *SugarPercept* variable to a corresponding one in the **mind** for making the agent aware of that information. For reasons of clarity, we introduce an additional variable supporting reasoning that stores the direction towards the highest amount of *CurrentSugar* out of the perceived ones. This variable is set in the *update_{mind}* function. The domain of the state variable *Direction* is $\{west, south, east, north, 2west, 2south, 2east, 2north\}$ (for the agent which's **body** has a perception radius of 2).

The action-selection function selects the action of moving into the optimal direction: $Act_m = \{Noop, West, South, East, North, 2West, 2South, 2East, 2North\}$ encoding direction and speed. In this simple version of the Sugarscape model, the action that the **mind** selects is also the one that the effectors of the **body** send to execution to the cell on which the **body** is located. The cell on which the **body** is located, will then move the **body** (together with its **mind**) to the envisioned cell, if the cell is not occupied.

3.3 Putting Elements together

Combining structural information and dynamics, the dynamics around **minds** and **bodys** must be more elaborated: A model of a **body** contains thus not only the state and the initial state, the set of actions that it can execute, but also the particular *sense* and *act* functions:

$$b_i = \langle \sigma_{b_i}, s_{b_i}(t_0), ExecAct_{b_i}, sense_{b_i}, act_{b_i} \rangle$$

Consequently, a similar collection can be done for the **minds**:

$$m_j = \langle \sigma_{m_j}, s_{m_j}(t_0), Act_{m_j}, perceive_{m_j}, update_{m_i}, action_{m_i} \rangle$$

The full description of a **region** contains the information of the state of the **region** a characterisation of its spatial representation, as well as a map-specific function for localisation. Then, there are three functions describing dynamical processes under control of the **region**: the independent dynamics of the **region**, the update of the **body** states depending on the **region** and the effect function executing the agents' actions.

$$r_x = \langle b_{r_x}, map_{r_x}, locate_{r_x}, updateRegion_{r_x}, updateBodyState_{r_x}, effect_{r_x} \rangle$$

So, a MABS model consists of a population of **bodys** without and with a **mind** and a representation of the spatial structures determining the basic structure of the shared environment. Thus, the population of entities consists of all **bodys**, all **minds** and the *embody* relation connecting them: $POP = \langle B, M, embody \rangle$ with $B = \{b_1, \dots, b_n\}$

the set of all **bodys**, and $M = \{m_1, \dots, m_k\}$ the set of all **minds**. Similarly, we collect all information on the environmental structure in $ENV = \langle R, C, connect \rangle$, combining all **regions** and their connections, that means the elements that describe how a number of **regions** are connected and the function that actually maps them together. The overall model is then defined combining population, environment and the *locate*-function connecting the **bodys** and the **regions**: $MABM = \langle POP, ENV, locate \rangle$. The overall *locate*-function has the **region**- specific *locate*-functions as elements.

This overall division into population and environmental structure reduces the environmental model of the MABS model to an active spatial structure. This is at first sight a little bit counter-intuitive and contradicts e.g. [19] who argue that the simulated environment may contain also explicit static entities. In our conceptualization the static elements are conceived as **bodys** without **minds** and are thus part of the population. We intentionally decided for this as a distinction between core agents and environmental agents is somehow artificial, especially if non-agent resources may have significant, autonomous dynamics.

4 Discussion and Conclusion

In this contribution, we proposed a first version of AMASON, an abstract core meta-model for MABS. Our goal was to suggest a meta-model that is generic enough to be able to capture a wide variety of agent-based simulation models. This idea poses a number of requirements, but also fixed starting points for our considerations: the concepts of agents shall be as simple as possible, but powerful enough to capture all forms of agents. Our basic agent concept is inspired by the *hysteretic* agents of Genesereth and Nilsson [11]; we wanted to give its internal state a structure in form of state variables in DEVS ([33]) as this is a clear and powerful generic concept. The distinction between **body** and **mind** is the result of a long discussion on how to capture the differences between passive objects or resources and active agents that we think must be made explicit providing ontological support. The structure of the environment in form of interconnected regions originates from a generalization of spatial representations observed in various MABS models. Interconnected regions are at first sight rather complex, yet this concept subsumes representations ranging from a single regions with a grid map to a network without any metric overhead. Additionally, it allows formulating environmental heterogeneity beyond heterogeneous populations.

AMASON is intentionally very simple, and in its current state covers only very basic structures and processes. Our focus was not on providing a highly elaborated meta-model with specific suggestions to all possibly occurring concepts, but one that fits best a broad variety of models in different domains, social science simulation and beyond. It contributes particularly to a clear conceptualization of embodiment and a generic spatial structure that can accommodate different types of spatial maps providing a heterogeneous environment for agent activities. Although AMASON is on high level of abstraction, we argue that it helps to clarify design decisions such as the granularity of actions that one has to take while designing a model.

AMASON is more basic than the AOSE meta-models mentioned above, which are more refined as they focus on providing languages for capturing interactions, coordination and organizations. On the other hand, our meta-model abstracts from particular

data types or basic implementation-specific aspects that tool or programming language specific meta-models would include. A previous version of our meta-model was partially inspired by [16] with its focus on interaction between agent and its environment. AMASON is more basic and should also work for models in which the environmental dynamics model is not so essential.

In its current status, AMASON can already be used for “testing” platforms aligning their concepts to this meta-model, clarifying where the platforms deviated from the core agent ideas for providing easy to use tools. Moreover, with the definitions of AMASON, we argue that the teaching of MABS becomes easier, as the core concepts are more clear and independent from any particular tool.

Naturally, there are a number of aspects that we have not yet tackled, but are important for MABS. In addition to the context information, the main point is the missing explicit conceptualization of direct interactions between agents, agent relations, and organizational structures. This is clearly an extension of the mind that we will address in the future. The conceptualization of basic interaction is also not as concise and fully clear as it could be, especially with respect to interactions that influences multiple regions. This forms a problem that we have to solve as the next step.

References

1. Bandini, S., Manzoni, S., Vizzari, G.: Agent based modeling and simulation: An informatics perspective. *J. Artificial Societies and Social Simulation* 12(4) (2009)
2. Bernon, C., Cossentino, M., Gleizes, M.-P., Turci, P., Zambonelli, F.: A study of some multi-agent meta-models. In: Odell, J., Giorgini, P., Müller, J.P. (eds.) *AOSE 2004*. LNCS, vol. 3382, pp. 62–77. Springer, Heidelberg (2005)
3. Beydoun, G., Low, G., Henderson-Sellers, B., Mouratidis, H., Gomez-Sanz, J.J., Pavón, J., Gonzalez-Perez, C.: *FAML: A generic metamodel for MAS development*. *IEEE Transactions on Software Engineering* 35(6), 841–863 (2009)
4. Dinu, R., Stratulat, T., Ferber, J.: A formal approach to MASQ. In: van der Hoek, W., Kaminka, G.A., Lespérance, Y., Luck, M., Sen, S. (eds.) *9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, Toronto, CA, pp. 1443–1444 (May 2010)
5. D’Inverno, M., Luck, M.: *Understanding Agent Systems*, 2nd edn. Springer (2004)
6. Epstein, J.M., Axtell, R.: *Growing Artificial Societies - Social Science from the Bottom Up*. The MIT Press (1996)
7. Ferber, J., Gutknecht, O.: A meta-model for the analysis and design of organizations in multi-agent systems. In: *ICMAS 1998: Proceedings of the 3rd International Conference on Multi Agent Systems*, pp. 128–135. IEEE Computer Society, Washington, DC (1998)
8. Ferber, J., Müller, J.-P.: Influences and reactions: a model of situated multiagent systems. In: *Proc. of the ICMAS 1996*. AAAI Press (1996)
9. Fuentes-Fernandez, R., Galan, J.M., Hassan, S., Lopez-Paredes, A., Pavon, J.: Application of model driven techniques for agent-based simulation. In: *Advances in Practical Applications of Agents and Multiagent Systems, PAAMS 2010*, Salamanca, Spain (April 2010)
10. Garro, A., Russo, W.: easyABM: A domain-expert oriented methodology for agent-based modeling and simulation. *Simulation Modelling Practice and Theory* 18, 1453–1467 (2010)
11. Genesereth, M.L., Nilsson, N.J.: *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann (1987)
12. Ghorbani, A., Bots, P., Dignum, V., Dijkema, G.: MAIA: a framework for developing agent-based social simulations. *Journal of Artificial Societies and Social Simulation* 16(2), 9 (2013)

13. Gilbert, N.: *Agent-based Models*. Sage Publications (2008)
14. Grimm, V., et al.: A standard protocol for describing individual-based and agent-based models. *Ecological Modelling* 198, 115–126 (2006)
15. Hahn, C., Madrigal-Mora, C., Fischer, K.: A platform independent meta model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems* 18(2), 239–266 (2009)
16. Helleboogh, A., Vizzari, G., Uhrmacher, A., Michel, F.: Modeling dynamic environments in multi-agent simulation. *Autonomous Agents and Multi-Agent Systems* 14(1), 87–116 (2007)
17. Hocaoglu, M.F., Firat, C., Sarjoughian, H.S.: Devs/rap: Agent-based simulation. In: *Proc. of the 2002 AIS Conference (AI, Simulation and Planning in Highly Autonomous Systems)*, Lisboa (April 2002)
18. Klügl, F.: *Towards a formal framework for multi-agent simulation models*. Technical Report 412, Institute of Computer Science, University of Würzburg (2007)
19. Klügl, F., Fehler, M., Herrler, R.: About the role of the environment in multi-agent simulations. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) *E4MAS 2004*. LNCS (LNAI), vol. 3374, pp. 127–149. Springer, Heidelberg (2005)
20. Livet, P., Muller, J.-P., Phan, D., Sanders, L.: Ontology, a mediator for agent-based modeling in social science. *Journal of Artificial Societies and Social Simulation* 13(1), 3 (2010)
21. Macal, C.M., North, M.J.: Tutorial on agent-based modelling and simulation. *Journal of Simulation* 4, 151–162 (2010)
22. Müller, J.-P.: *Towards a formal semantics of event-based multi-agent simulations*. In: David, N., Sichman, J.S. (eds.) *MAPS 2008*. LNCS, vol. 5269, pp. 110–126. Springer, Heidelberg (2009)
23. North, M.J., Macal, C.M.: *Managing BusinessComplexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation*. Oxford University Press (2007)
24. Oechslein, C.: *Vorgehensmodell mit integrierter Spezifikations- und Implementierungssprache zur Multiagentensimulation*. PhD thesis, Computer Science Faculty, University of Würzburg, Germany (2003)
25. Omicini, A., Ricci, A., Viroli, M.: Artifacts in the a&a meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems* 17, 432–456 (2008)
26. Parker, M.T., North, M.J., Collier, N.T., Howe, T.R., Vos, J.R.: Agent-based meta-models. In: Sallach, D.L., Macal, C.M., North, M.J. (eds.) *Proc. the Agent 2006 Conf. on Social Agents: Results and Prospects*, pp. 69–79 (2006)
27. Railsback, S.F., Grimm, V.: *Agent-based and Individual-based Modeling: A Practical Introduction*. Princeton University Press (2011)
28. Siegfried, R., Lehmann, A., Khayari, R.E.A., Kiesling, T.: A reference model for agent-based modeling and simulation. In: *Proceedings of SpringSim 2009, The 2009 Spring Simulation Multiconference*, pp. 23:1–23:8 (2009)
29. Uhrmacher, A.M.: Object-oriented and agent-oriented simulation-implications for social science applications. In: Doran, J., Gilbert, N., Müller, U., Troitzsch, K.G. (eds.) *Social Science Micro Simulation*, pp. 432–447. Springer, Berlin (1996)
30. Urban, C.: PECS: A reference model for the simulation of multi-agent systems. In: Suleiman, R., Troitzsch, K.G., Gilbert, N. (eds.) *Tools and Techniques for Social Science Simulation*, pp. 83–114. Physica-Verlag (2000)
31. Weyns, D., Holvoet, T.: A formal model for situated multi-agent systems. *Fundamenta Informaticae* 63(2-3), 125–158 (2004)
32. Wooldridge, M.J.: *Introduction to Multiagent Systems*, 2nd edn. John Wiley & Sons (2009)
33. Zeigler, B.P.: *Object Oriented Simulation With Hierarchical Modular Models: Intelligent Agents and Endomorphic Systems*. Academic Press (1990)

Robust PI - The Simulation Robust Library Component for Pioneer 3DX

Konrad Kułakowski and Piotr Matyasik

Department of Applied Computer Science,
AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Cracow, Poland
{konrad.kulakowski,piotr.matyasik}@agh.edu.pl

Abstract. The article presents *RobustPI* - an extension of the *Robust* middleware library, which allows the widely recognized *Pioneer 3-AT* robot to be programmed as a *Webots* based virtual hardware. Like *Lego Mindstorms NXT* robots, the *Pioneer 3-AT* model and *Webots* simulator are very popular as experimental platforms in many places all over the world. For this reason, the authors hope that the extension *RobustPI* may also be of interest and use for many people involved in carrying out research and teaching in the field of intelligent robotics.

Keywords: intelligent autonomous robots, the Robust library, robotic middleware.

1 Introduction

The universality of intelligent autonomous robots in everyday practice is a fact. Many people around the world use autonomous robotic vacuum cleaners like *Roomba*TM, self-propelled autonomous lawn mowers or intelligent parking assist systems. All of these constructions are sophisticated combinations of hardware and software. The success of the robotic construction depends on the hardware reliability, software quality and how well both parts can work together. The cost of developing an intelligent robot of a commercial quality can be significant¹. Besides creating hardware, the process of achieving high quality robotic software requires a lot of prototyping, testing and a number of experiments. The aim of the research phase is to refine the robot control algorithms and tuning cooperation between the hardware platform and the software layer. Since for many reasons it is not always possible to provide software engineers with appropriate robotic hardware, robotic simulation environments are very popular. One of the widely recognized simulators is the *Webot*TM simulator². It supports a number of popular robotic constructions including *Pioneer 3-AT* or *Nao Robot*³.

The proposed extension of the *Robust* library is a step towards opening this library to the simulation environment. *RobustPI*, the newly created *Robust* module, allows the user to create software which is executable on the *Pioneer* virtual

¹ iRobot Corporation R&D Expense report (available online at ycharts.com).

² www.cyberbotics.com

³ www.aldebaran-robotics.com

robot, running under the *Webots* environment. The current work shows the location of the *RobustPI* module in the general architecture of the *Robust* library and tries to put it into a broader 3-tier architecture context (Sec. 2.2 and 3). In addition to the description of the module design (Sec. 3.2) and functionality (Sec. 3.3) the work also contains a brief example of an application that uses the proposed extension (Sec. 4).

2 The Robust Library Architecture

2.1 Architectural Perspective

One of the popular architectural styles in intelligent robotic software is state-hierarchy architecture. In this approach, activities are organized by the scope of time knowledge [16]. A good example of such architecture is 3T, or 3-Tiered, mobile robot architecture. The layers lie one above the other. The top, deliberative, layer deals with planning, the bottom layer is purely reactive and corresponds to the robot's skills, and the middle layer is designed to mediate between the two outer layers. The popularity and flexibility of this approach are confirmed by numerous examples [2,7,6,17,4,9]. Because of the specificity of the processing, similar to that in the Albus reference Model [1], different layers have different time perspectives. The top deliberative layer is responsible for planning the robot activities. Hence, it has the the widest time perspective. Usually, it does not need to make a decision immediately and may take a while to think. At the decision-making level very often robotic architectures use the theory and practice of agent systems [15,20,18]. In this approach, decision-making processes might be supported by different methods including formal logic [10], neural networks [19], decision trees [5] and others. On the opposite side is the bottom layer. It should react as quickly as possible. Therefore, there is a common tendency to implement it close to the hardware [8,4]. The purpose of the middle layer is to interact between the low level reactive procedures (very often equated with skills) and the deliberative abstract logic responsible for planning. Its job relies on behavior generation understood as establishing the sequence of primitive skills forming some specific behaviors.

The *Robust* library is designed as a reactive layer acting between the software system and the robotic hardware. An example of using the *Robust* library together with a behavior generation layer is [14]. In that example, the robot behavior has been formally modeled with the help of *Concurrent Communicating List (CCL)* notation [12]. The presented *RobustPI* module can be considered as an extension of the *Robust* based reactive level. The *Robust* library has been designed in such a way so as not to be tied to any particular behavior layer implementation. It remains open source and is intended to be compatible with other hardware and software solutions.

2.2 Implementation Perspective

The primary reason for building the *Robust* library [11] was to provide a reliable method for writing advanced control Java programs running on the *Mindstorms*TM

NXT intelligent brick. Since the brick has little RAM, such programs need to be executed on a personal computer, whilst the sensor readings and the control commands are transmitted wirelessly. Thus, at the beginning, the library consisted of two components, *RobustNXT* and *RobustPC*. The first one was designed as an embedded application running on the brick and was responsible for handling communication messages and performing low level control operations, whilst the second was a Java library providing the users the high level control robot API. Over time, the library has gained new modules (Fig. 1) allowing the user to control the robot *Hexor II (RobustHX)* [13], and to program a virtual *Pioneer 3-AT* robot running under the control of the *Webots*TM simulator (*RobustPI*).

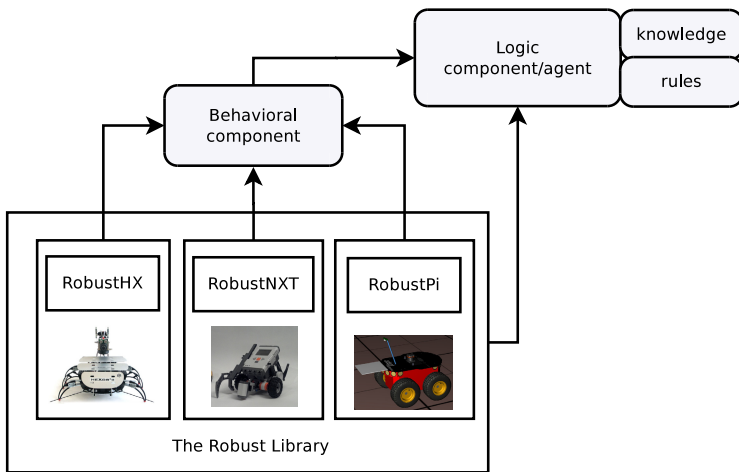


Fig. 1. The Robust library architecture - 3T perspective

3 The RobustPI Component

The *RobustPI* component was designed as the third hardware implementation in the Robust family. It is unique because it is not a physical robot but a virtual one. Below is a short description of its architecture.

3.1 Overall Control Layout

The *RobustPI* component is designed as a pair composed of the virtual robotic hardware embedded into the *Webots*TM environment and the standard *Robust* library interface adapter. The first part of the component is implemented as an embedded application running as a *Webots*' robot controller. It is responsible for performing two tasks: reading new actuator settings from the *Robust* library adapter, and sending the latest sensor readings including positions of the actuators to the *Robust* library adapter.

The second part of the *RobustPI* component - the *Robust* library interface adapter is a separate software module, which actually implements the *RobustAPI*. Both parts communicate with each other via a TCP network socket, which makes it possible to deploy the *Webots*TM simulator and the control application on separate computers (see Fig. 2).

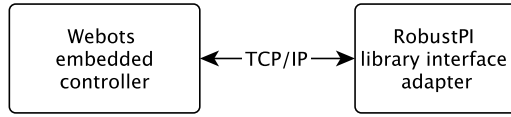


Fig. 2. The *RobustPI* component scheme

3.2 The Low-Level Design

The *Robust* library interface adapter continuously communicates with its embedded counterpart. It receives actual information about the robot state and stores it locally in the form of a *Pioneer3ATState* class object. Hence, when the *Robust* API is queried for the robot configuration data, the *Pioneer3ATState* local cache is used. The cache update rate depends on the *Webots* controller program and the configuration. Although no historical data are provided by the *Robust API* itself, similar functionality can be easily implemented on the top of the *Robust* library interface adapter.

In the development stage, there is another more sophisticated embedded part of the *RobustPI* component, based on the *Supervisor* controller. It allows the user to access some internal simulation data which may be indispensable during automatic evaluation of certain aspects of the control algorithm.

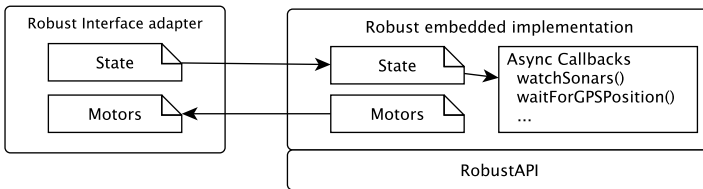


Fig. 3. *RobustPI* - the command processing scheme

An important group of the *Robust* interface methods are those that allow the user to specify subsequent robot moves [11]. Each of these commands creates a new *Motor* object. After its creation, such an object contains data for servos according to the selected movement type, parameters and the robot's geometry. When created, it is serialized, and sent over TCP/IP to the *Webots* controller. Afterwards the new settings are applied to the actuators.

To minimize network traffic, all the sensors and servos readings are transferred asynchronously. Hence, in order to receive notification that the specified parameters or readings have been changed, the user needs to implement appropriate listener objects and specify what kind of changes should trigger the listeners (Fig. 3).

3.3 Interface

The *Robust* API can be split into three groups wherein each of these groups provides synchronous and asynchronous interface methods (see Tbl. 1). The first group has only one member (*System*) and is used mainly for initialization, troubleshooting, general information and disconnecting from the hardware unit. An asynchronous API provides the ability to add a logging facility to the robot for debugging purposes.

Table 1. The Robust API overview

Group	API Name	Async
Special purpose	System	✓
Commands	Move	✓
	MoveLegged	✓
	Head	✓
Readings	Light	✓
	Laser	✓
	IR	✓
	Touch	✓
	Sonar	✓
	SonarFixed	✓
	Compass	
GPS		

The next *API* group (*Commands*) gathers functions related to the robot movement. There include the *Move* interface, which covers differential wheeled robots like *Pioneer*⁴ or *Khepera*⁵. It allows the robot to move forward and backward, turn in place, and drive along a specified arc. The second interface within this group is *MoveLegged*. It was designed to control the *Hexor II* robot⁶. Due to problems with distance and angle tracking for particular steps on different surfaces, unlike the *Move* interface, it does not provide distance or angle as movement parameters. The last interface in this group, the *Head* interface, controls the *Hexor II* head. The asynchronous methods defined within this group

⁴ <http://www.mobilerobots.com>

⁵ <http://www.k-team.com>

⁶ <http://www.stenzel.com.pl>

allow registration of functions to be executed upon completion of the currently processed movement command.

Third, the most numerous API group provides interfaces for accessing a variety of actual and virtual hardware. The *Light* interface is for reading luminosity, and the *Laser* interface for handling different scanning laser rangefinders. It also includes the *Infrared* interface, which is responsible for simple proximity infrared sensor group handling. It provides a vector indicating which specific sensor was triggered. Connecting a particular table index with the position of the detector on the robot body is left to the user. There is also the *Touch* interface providing access to simple mechanical bumpers, and the *Sonar* and *SonarFixed* interfaces that covers the functionality of ultrasonic range finders. The *Robust* API provides the user with an abstraction layer that unifies units and re-calculates the values of arguments of methods according to the robot settings.

4 Sample Algorithm Excerpt

To illustrate the usage of the *Robust* library and the *RobustPI* module, let us consider the following sample algorithm. Inspiration for the algorithm was *Valentino Braitenberg's* idea concerning vehicles and synthetic psychology [3]. Hence, according to one of his experiments, the robot is attracted by an obstacle, and whenever it perceives obstacles it “attacks” them. In the presented modified version of this approach, if the robot approaches closer to the obstacle than a certain configurable “escape” distance, then it turns back and finds another target.

```

1 private void nowRunAwayFromTheObstacle(){
2     move.moveTurnInPlace(180,60);
3     move.waitForMoveChange();
4     move.moveForward(100, 20, 0);
5     move.waitForMoveChange();
6 }
```

Listing 1. Retreat from the obstacle

The retreat routine (Listing 1) is executed whenever a robot detects an obstacle. At the beginning, the robot is ordered to turn 180 degrees in a place with a given rotation speed (Line: 2), then the robot is asked to move 100 centimeters forward with a speed of 20 centimeters per second. The calls of *waitForMoveChange()* (Lines: 3, 5) suspend the algorithm execution until the currently processed move commands are completed. The *waitForMoveChange()* method is implemented as an asynchronous callback related to asynchronous *Sonar* and *Laser* interfaces (see Listing: 2). For the sake of simplicity, some non essential lines of the *nowRunAwayFromTheObstacle()* method code have been omitted. For instance, the code responsible for registering and unregistering proximity alarm listeners is not discussed here⁷.

⁷ The full code of the presented example is available at code.google.com/p/robust-nxt

```

7 sonarAsync.registerProximityAlarmListener(
8     new RobustAPISonarAsync.SonarProximityAlarmListner() {
9         public int getAlarmDistance() {return 20;}
10        public int getMode() {return 1;}
11        // object closer than alarmDistance
12        public void handleProximityAlarm(int distance) {
13            nowRunAwayFromTheObstacle();
14        }
15 });

```

Listing 2. Running away from the obstacle

Besides synchronous method calls (Listing: 1, Lines: 2 - 5), there is very often a need to subscribe to asynchronous events coming from the hardware platform. The method *registerProximityAlarmListener()* is an example of asynchronous communication between the *Robust* library and the *Java* control application (Listing: 2). The method gets as its input the new *SonarProximityAlarmListner* object overriding the following methods:

- *getAlarmDistance()* - returns proximity alarm threshold,
- *getMode()* - returns alarm mode (searching for obstacles closer than alarm distance),
- *handleProximityAlarm()* - callback executed when alarm becomes active.

Of course, multiple callback functions referring to a single event source can also be registered.

Although the high level control code samples presented above were written for the purpose of testing the *RobustPI* module, they actually do not depend on the hardware platform. Hence, similar code can also be successfully executed on the standard two-wheeled *Mindstorms NXT* vehicle. Of course, not all of the robotic constructions implement the same set of *Robust library* API methods. Thus, the library allows the user to dynamically determine which method is implemented. This functionality makes it possible to write flexible robust programs that will be able to run properly on even vastly different hardware platforms.

5 Summary

In this paper, *RobustPI* - the new extension of the *Robust* library for controlling *Pioneer 3DX* virtual hardware, has been presented. Using the *Webots*TM simulation environment shortens the time needed for preparing robotic experiments, and reduces the cost of developing new control algorithms. Although the *RobustPI* module tries as often as possible to re-use existing interfaces, it also defines new ones, such as *Laser* range finder API. The control algorithms built on the top of the *Robust* library are well separated from the actual hardware. Hence, they can be easily executed on physically different but functionally similar robots.

Although the *Robust* APIs are ready to use, a lot of aspects still need to be improved. In particular, the introduction of a hierarchy of interfaces grouped by similar functionality is becoming increasingly important. Further development of the presented solution will focus on the creation of a fully functional, formally verifiable behavior component. Research on an efficient and scalable deliberative component will also be conducted.

Acknowledgement. This research is supported by AGH University of Science and Technology, contract no.: 10.10.120.859.

References

1. Albus, J.S.: Outline for a theory of intelligence. *IEEE Transactions on Systems, Man, and Cybernetics* 21(3), 473–509 (1991)
2. Arkin, R.: Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Robotics and Autonomous Systems* 6(1-2), 105–122 (1990)
3. Braitenberg, V.: *Vehicles: Experiments in Synthetic Psychology*. The MIT Press (September 1984)
4. Brunete, A., Hernando, M., Gambao, E., Torres, J.E.: A behaviour-based control architecture for heterogeneous modular, multi-configurable, chained micro-robots. *Robotics and Autonomous Systems* 60(12), 1607–1624 (2012)
5. Farahnakian, F., Mozayani, N.: Learning through Decision Tree in Simulated Soccer Environment. In: *International Conference on Computational Intelligence and Security, CIS 2008*, pp. 68–70 (2008)
6. Fierro, R.B., Aveek, K.D., Spletzer, J.R., Esposito, J.M., Kumar, V., Ostrowski, J.P., Pappas, G.J., Taylor, C.J., Hur, Y., Alur, R., Lee, I., Grudic, G.Z., Southall, B.: A framework and architecture for multi-robot coordination. *International Journal of Robotic Research* 21(10-11), 977–998 (2002)
7. Firby, R.J., Kahn, R.E., Prokopowicz, P.N., Swain, M.J.: An architecture for vision and action. In: *Fourteenth International Joint Conference on Artificial Intelligence*, pp. 72–79 (1995)
8. Gat, E.: On three-layer architectures. *Artificial Intelligence and Mobile Robots* (1997)
9. Hoshino, K., Kunii, Y.: Three-layered architecture for tele-operator and its system test. In: Kim, J.-H., Matson, E.T., Myung, H., Xu, P. (eds.) *Robot Intelligence Technology and Applications*. AISC, vol. 208, pp. 105–114. Springer, Heidelberg (2013)
10. Klimek, R.: A Deduction-based System for Formal Verification of Agent-ready Web Services. In: *Advanced Methods and Technologies for Agent and Multi-Agent Systems*. *Frontiers of Artificial Intelligence and Applications*, vol. 252, pp. 203–212. IOS Press (2013)
11. Kułakowski, K.: Robust - Towards the Design of an Effective Control Library for Lego Mindstorms NXT. In: *Proceedings of Conference on Software Engineering Techniques CEE-SET* (September 2009)
12. Kułakowski, K.: Concurrent systems modeling with CCL. *Automatyka* (2012)
13. Kułakowski, K., Matyasik, P.: RobustHX - The Robust Middleware Library for Hexor Robots. In: Ando, N., Balakirsky, S., Hemker, T., Reggiani, M., von Stryk, O. (eds.) *SIMPAN 2010*. LNCS, vol. 6472, pp. 241–250. Springer, Heidelberg (2010)

14. Kułakowski, K., Szmuc, T.: Modeling Robot Behavior with CCL. In: Noda, I., Ando, N., Brugali, D., Kuffner, J.J. (eds.) SIMPAR 2012. LNCS, vol. 7628, pp. 40–51. Springer, Heidelberg (2012)
15. Low, K., Leow, W., Ang Jr., M.: A hybrid mobile robot architecture with integrated planning and control. In: Proceedings of International Conference on Autonomous Agents and Multiagent Systems, pp. 219–226. ACM Press (2002)
16. Murphy, R.: Introduction to AI robotics, p. 466. MIT Press (January 2000)
17. Palomeras, N., El-Fakdi, A., Carreras, M., Ridao, P.: Cola2: A control architecture for auvs. *IEEE Journal of Oceanic Engineering* 37(4), 695–716 (2012)
18. Paraschos, A., Spanoudakis, N.: Model-Driven Behavior Specification for Robotic Teams. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, AAMAS (2012)
19. Peng, L., Liu, H.Y.: Decision-Making and Simulation in Multi-Agent Robot System Based on PSO-Neural Network. In: IEEE International Conference on Robotics and Biomimetics, ROBIO 2007, pp. 1763–1768 (2007)
20. Xiao, W., Liu, T., Baltés, J.: An intuitive and flexible architecture for intelligent mobile robots. In: The Second International Conference on Autonomous Robots and Agents(ICARA), Palmerston North, pp. 52–57 (2004)

Secrecy Preserving BDI Agents Based on Answer Set Programming

Patrick Krümpelmann and Gabriele Kern-Isberner

Technische Universität Dortmund, Information Engineering Group

Abstract. We consider secrecy from the point of view of an autonomous knowledge-based and resource-bound agent with incomplete and uncertain information, situated in a multi agent system. We investigate properties of secrecy and the preservation thereof in this setting and formulate desirable properties. Based on these ideas we develop a flexible BDI-based agent model and define an instance widely based on answer set programming. We show that and how our model and instance satisfy the proposed properties. We implemented our developed extendable framework for secrecy-preserving agents based on JAVA and answer set programming.

1 Introduction

On the topic of secrecy a large body of work exists and diverse definitions of secrecy in various settings with different properties have been developed. For multiagent systems the main research focus herein lies on strong notions of secrecy of a whole (multiagent) system, for an overview see [4,11]. Secrecy is generally imposed by some global definition of secret information from a global, complete view of the entire system. While substantial work on the definition of secrecy exists mechanisms for secrecy preservation in multiagent systems are lacking.

In this work we consider secrecy and secrecy preservation from the point of view of an autonomous knowledge-based agent with incomplete and uncertain information, situated in a multiagent system. Agents reason under uncertainty about the state of the environment, the reasoning of other agents and possible courses of action. They pursue their goals by performing actions in the environment including the communication with other agents. On the one hand, the exchange of information with other agents is often essential for an agent in order to achieve its goals; especially if the agent is part of a coalition. On the other hand the agent is interested, or obliged, not to reveal certain information, its secrets. Restriction of communication leads to a loss of performance and utility of the individual agent, coalitions and the whole multiagent system. A good solution of the implied conflict between the agent's goal to preserve secrecy and its other goals is one that restricts communication as little as necessary in order to preserve secrecy. Secrecy of information and in particular the inference problem depend on the representation of information and the appropriate modeling of background information and of the reasoning capabilities of the agents.

Our contributions lay in several aspects. We investigate, motivate and formalize novel and general properties of an agent model with respect to secrecy and secrecy preservation from a subjective perspective of an agent with incomplete information. We develop an epistemic agent model for secrecy preservation, which is based on the abstract model presented in [7]. We show that besides the pure declaration of secrets, the properties of the belief change, the attacker modeling and the means-end reasoning components of the agent are essential for secrecy declaration and preservation and define the properties of each of the three components in detail. Moreover we define answer set programming (ASP) [3] based concrete instances to illustrate how the properties can be satisfied. We implemented the general framework as well as the ASP instance presented in this work using JAVA and available ASP solvers.

The remainder of this paper is structured as follows. First we give a very brief introduction to ASP in Section 2. Then, in Section 3, we motivate and informally develop desiderata of a secrecy preserving agent based on the belief change, the attacker modeling and the means-end-reasoning component of an agent. Based on these ideas we formalize our notion of an epistemic agent in Section 4. In Section 5 we elaborate the first, the belief change, component of an agent with respect to secrecy. In Section 6 we elaborate the second component by presenting a formalization and an approach to attacker modeling and its relevance for secrecy preservation. In Section 7 we consider the third component and develop properties and for means-end-reasoning and how to satisfy them in our instance. In Section 8 we sum up, discuss the relation to other approaches and give an outlook.

2 Answer Set Programming Basics

We give a brief introduction to answer set programming [3]. Let At be the set of all atoms and Lit the set of all literals $Lit = At \cup \{\neg A \mid A \in At\}$. A rule r is written as $H(r) \leftarrow \mathcal{B}^+(r), \mathcal{B}^-(r)$, the head of the rule $H(r)$ is either empty or consists of a single literal, the body consists of $\mathcal{B}^+ = \{L_0, \dots, L_m\}$ and $\mathcal{B}^- = \{not L_{m+1}, \dots, not L_n\}$ with $L_0, \dots, L_n \in Lit$. The language of rules constructed over the set of atoms At is referred to as \mathcal{L}_{At}^{asp} . A finite set of sentences from \mathcal{L}_{At}^{asp} is called an extended logic program $P \subseteq \mathcal{L}_{At}^{asp}$. A state S is a set of literals that does not contain complementary literals L and $\neg L$ is called. A state S is a model of a program P if for all $r \in P$ if $\mathcal{B}(r)^+ \subseteq S$ and $\mathcal{B}(r)^- \cap S = \emptyset$ then $H(r) \cap S \neq \emptyset$. The reduct P^S of a program P relative to a set S of literals is defined as $P^S = \{H(r) \leftarrow \mathcal{B}^+(r) \mid r \in P, \mathcal{B}^-(r) \cap S = \emptyset\}$. An answer set of a program P is a state S that is a minimal model of P^S . The set of all answer sets of P is denoted by $AS(P)$. Rule schemas can use variables which we denote by x, y, z and $_$ for the anonymous variable [3].

3 Properties of Secrecy and Secrecy Preservation

In this section we argue that the definition of secrecy is complex and dependent on various aspects which influence the actually obtained secrecy and the

restriction of information flow. Furthermore, we elaborate the key ideas and properties of secrecy preserving agents. In the following we give the introduction to our running example and then statements about secrecy followed by examples.

Example 1. Consider an employee, *emp*, working in a company for his boss *boss*. He wants to attend a strike committee meeting (*scm*) next week and has to ask his boss for a day off in order to attend. It is general knowledge that the agent *boss* puts every agent who attends the *scm* on her blacklist of employees to be fired next.

Secrets are not uniform in their content as an agent has different secrets with respect to different agents.

Example 2. In our example, *emp* wants to keep his attendance to the *scm* secret from *boss* but not from other employees that also want to attend the *scm*.

Secrets are also not uniform with respect to their strength. That is, an agent wants to keep some information more secret than other. These differences in strength of secrets arise naturally from the value of the secret information. The value of secret information depends on the severeness of the negative effects, or the cost, for the agent resulting from disclosure of the secret information. These costs can differ widely and consequently the agent is interested in not revealing secret information to different degrees.

Example 3. *emp* does not even want his *boss* to be suspicious about him attending the *scm* (secrecy with respect to a credulous reasoner). He also does not want other employees that are against the strike to know that he attends the *scm*. However, with respect to the latter he considers it sufficient that they do not know for sure that he attends (secrecy with respect to a skeptical reasoner).

Secrets are also not static, they arise, change and disappear during runtime of an agent such that it has to be able to handle these changes adequately.

Example 4. If *emp* realizes that *boss* overheard his phone call with the strike committee he should give up his corresponding secret.

These considerations lead to the following formulation of properties of secrets: (*S1*) secrets can be held with respect to specific agents, (*S2*) secrets can vary in strength, (*S3*) secrets can change over time.

Now we want to formulate properties of a secrecy preserving agent and begin with an informal formulation. We assume a multiagent system with a set of agents \mathfrak{A} . We use the agent identifier \mathcal{X} to denote an arbitrary agent. For the representation of the secrecy scenario it is convenient to focus on the communication between two agents, the modeled agent \mathcal{D} which wants to defend its secrets from a potentially attacking agent \mathcal{A} . Defining secrets does not define the preservation of secrecy and its properties. The intuitive formulation of our notion of secrecy preservation can be formulated as: *An agent \mathcal{D} preserves secrecy if, from its point of view, none of its secrets Φ that it wants to hide from agent \mathcal{A} is, from \mathcal{D} 's perspective, believed by \mathcal{A} after any of \mathcal{D} 's actions (given that \mathcal{A} does not believe Φ already).*

The actual quality of secrecy preservation is highly dependent on the accuracy of the view of \mathcal{D} on the agent \mathcal{A} and its supposed reasoning capabilities as well as on \mathcal{D} 's information processing and adaptation of its beliefs and view on \mathcal{A} in the dynamic scenario. To make the importance clear, a completely ignorant agent would never subjectively violate secrecy as it would ignore its violation of secrecy. Likewise underestimating as well as overestimating the capabilities of an \mathcal{A} can lead to a violation of secrecy. In particular a secrecy preserving agent should satisfy the following properties: (P1) The agent is aware of the information communicated to other agents and the meta-information conveyed by its actions, (P2) The agent simulates the reasoning of other agents, (P3) The agent considers possible meta-inferences from conspicuous behavior such as (a) selfcontradiction, (b) refusal, (P4) For all possible states and perceptions the agent does not perform any action that leads to secrecy violation, (P5) The agent only weakens secrets if it is unavoidable due to information coming from third parties and only as much as necessary.

As we shall see, the properties (P1) and (P5) are related to the belief change component of \mathcal{D} , (P2) and (P3) to the way \mathcal{D} models \mathcal{A} and (P4) to the means-end reasoning behavior of \mathcal{D} . In the following we elaborate on all properties, formalize them and develop corresponding agent components and show which formalized properties are satisfied.

4 Formal Framework

We present an epistemic model of agency which stresses the knowledge representation and reasoning under uncertainty and incorporates secrets, and views of an agent on the information available to other agents. The reasoning under uncertainty is formalized by belief operators which can be more or less credulous. We then use these notions to define secrets and secrecy preservation.

The general framework as presented in [7] generalizes a variety of agent models. Here, we use a more concrete model loosely based the well known beliefs, desires, intentions (BDI) architecture [10]. Note that the BDI model just serves as an example agent model and that all properties and operators developed here are independent of it and are applicable to virtually all agent models. In our epistemic view of agency, the agent's epistemic state contains a representation of its current desires and intentions which guides its behavior. The functional component of a BDI agent consists of a change operation of the epistemic state and an action function, executing the next action as determined by the current epistemic state. Our agent model is illustrated in Figure 4.

Definition 1 (Epistemic BDI Agent). *An agent \mathcal{D} is a tuple $(\mathcal{K}_{\mathcal{D}}, \xi_{\mathcal{D}})$ comprising an epistemic state $\mathcal{K}_{\mathcal{D}}$ and a functional component $\xi_{\mathcal{D}}$. A BDI-Epistemic-State is a tuple $\mathcal{K}_{\mathcal{D}} = \langle \langle V_{\mathcal{D},W}, \mathcal{V}_{\mathcal{D}}, \mathcal{S}_{\mathcal{D}} \rangle, \Delta_{\mathcal{D}}, \mathcal{I}_{\mathcal{D}} \rangle$. It consists of a world view $V_{\mathcal{D},W}$, a set of agent views $\mathcal{V}_{\mathcal{D}} = \{V_{\mathcal{D},X} \mid X \in \mathfrak{A} \setminus \{\mathcal{D}\}\}$, a set of secrets $\mathcal{S}_{\mathcal{D}}$, a set of desires $\Delta_{\mathcal{D}}$, and a set of intentions $\mathcal{I}_{\mathcal{D}}$. We refer to the first component as the agent's beliefs $\mathcal{B}(\mathcal{K}_{\mathcal{D}}) = \mathcal{B}_{\mathcal{D}}$. We set $V_W(\mathcal{B}) = V_W(\mathcal{K}_{\mathcal{D}}) = V_{\mathcal{D},W} \subseteq \mathcal{L}_{At}^{asp}$, $V_X(\mathcal{B}_{\mathcal{D}}) = V_X(\mathcal{K}_{\mathcal{D}}) = V_{\mathcal{D},X} \subseteq \mathcal{L}_{At}^{asp}$ and $\mathcal{S}(\mathcal{K}_{\mathcal{D}}) = \mathcal{S}(\mathcal{B}_{\mathcal{D}}) = \mathcal{S}_{\mathcal{D}}$. The functional*

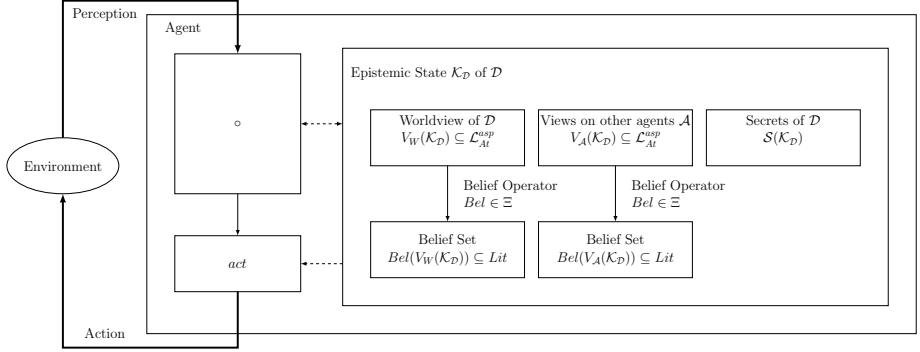


Fig. 1. Epistemic Agent Model

component $\xi_{\mathcal{D}} = (\circ_{\mathcal{D}}, \text{act}_{\mathcal{D}})$ consists of an change operator $\circ_{\mathcal{D}}$ and an action operator $\text{act}_{\mathcal{D}}$.

A belief operator determines the currently held beliefs of the agent given a view. In the ASP setting beliefs are represented by an answer set, i. e. a set of literals, and a view by an extended logic program. An agent with incomplete and uncertain information might employ different belief operators which are more or less credulous. A belief operator is *more credulous* than another one if for all views the belief set of the latter is a subset of the belief set of the former.

Definition 2 (Belief Operators). A belief operator is a function $Bel : \mathcal{L}_{At}^{asp} \rightarrow Lit$. Ξ is a finite family of belief operators Ξ plus the ignorant operator $Bel_{\emptyset}(V) = \emptyset$. We assume a credulity order $<$ on Ξ such that if $Bel < Bel'$ for some $Bel, Bel' \in \Xi$ then for all $V \in \mathcal{L}_{At}^{asp}$ $Bel(V) \subseteq Bel'(V)$. The ASP belief operator family is given by $\Xi^{asp} = \{Bel_{skip}^{asp}, Bel_{cred}^{asp}, Bel_{\emptyset}\}$, $Bel_{cred}^{asp}(P) = \cap AS(P)$ and $Bel_{skip}^{asp}(P) = \cup AS(P)$ and $Bel_{cred}^{asp} \succ Bel_{skip}^{asp} \succ Bel_{\emptyset}$.

The definition of a family of belief operators abstracts from the underlying formalism and inference mechanism. Thereby it captures a wide range of formalisms from purely qualitative ones to plausibilistic ones. The ASP instance considered here is just one example, used for the illustration of the approach in this paper. To define secrets, the information to be kept secret has to be defined. Also, the agent from which the information shall be kept secret has to be defined and lastly the strength of the secret has to be expressed. We make use of the belief operators to express the strength of a secret.

Definition 3 (Secrets). A secret is a tuple (Φ, Bel, \mathcal{A}) which consists of a formula $\Phi \in Lit$, a belief operator $Bel \in \Xi$ and an agent identifier $\mathcal{A} \in \mathfrak{A}$. The set of secrets of agent \mathcal{D} is denoted by $\mathcal{S}(\mathcal{K}_{\mathcal{D}})$.

Assigning a more credulous belief operator to a secret leads to a stronger protection of secret information, as illustrated in Example 3. That is, if \mathcal{D} reveals some

information, a credulous attacker might infer some secret information while a skeptical one with the same revealed information might not. In the former case the defender should not have revealed the information. Formally, considering two secrets (Φ, Bel, \mathcal{A}) and $(\Phi, Bel', \mathcal{A})$, the former is stronger than the latter iff $Bel \succ Bel'$.

Observation 1. *The definition of secrets in Definition 3 satisfies (S1), (S2) and (P2).*

Example 5. We model the *scm* scenario from Example 2 and in particular the initial epistemic state of the employee *emp*, $\mathcal{K}_{emp} = \{\{V_{emp,W}, \mathcal{V}_{emp}, \mathcal{S}_{emp}\}, \Delta_{emp}, \mathcal{I}_{emp}\}$, with $\mathcal{V}_{emp} = \{V_{emp,boss}\}$. We assume that *emp* and *boss* share the same background knowledge, such that $V_{emp,W} = V_{emp,boss} = P_{view}$ with:

$r_1 : \neg attend_work$	\leftarrow	<i>excused.</i>
$r_2 : excused$	\leftarrow	<i>attend_scm.</i>
$r_3 : excused$	\leftarrow	<i>medical_appointment.</i>
$r_4 : attend_scm$	\leftarrow	<i>not medical_appointment, asked_for_excuse.</i>
$r_5 : medical_appointment$	\leftarrow	<i>not attend_scm, asked_for_excuse.</i>
$r_6 : blacklist$	\leftarrow	<i>not excused, \neg attend_work.</i>
$r_7 : blacklist$	\leftarrow	<i>attend_scm.</i>
$r_8 : attend_work$	\leftarrow	<i>not \neg attend_work.</i>

The program encodes that *emp* has to be excused in order to not go to work (r_1). He is excused if he attends the scm or if he has a medical appointment (r_1 – r_2). If he asks to be excused these two possible explanations exist (r_4 – r_5). If he is absent without being excused he will be blacklisted (r_6). If he attends the scm, and is thus excused, he will still be blacklisted (r_7). He normally goes to work (r_8). The set of answer sets is $AS(P_{view}) = \{\{attend_work\}\}$. The secret of the employee is $\mathcal{S}_{emp} = \{(attend_scm, Bel_{skept}^{asp}, boss)\}$. The initial set of desires of the employee is $\Delta_{emp} = \{attend_scm\}$.

For secrecy preservation the dynamics of the epistemic state induced by actions and perceptions have to be considered. We assume a set of possible actions *actions* and a set of possible perceptions *percepts*, including the empty ones. To make the formalism more comprehensible and to illustrate a concrete instance we consider communicating agents here. Note that other types of actions and perceptions, such as manipulations in some environment are also captured by the general framework. For the illustration here, we assume that actions as well as perceptions τ are speech acts from a set of speech acts $\langle A_s, \{A_{r_1}, \dots, A_{r_n}\}, type, \Phi \rangle$ specifying the source $A_s \in \mathfrak{A}$, the receivers $A_{r_1} \in \mathfrak{A}$ to $A_{r_n} \in \mathfrak{A}$, the type *type* and the informational content $\Phi \in Lit$. The main difference between perceptions and actions is that perceptions represent actions performed by other agents while actions represent the actions the agent under consideration has performed. We differentiate between requesting speech acts $\Psi_R = \{\text{query, justify}\}$ and informative speech acts $\Psi_I = \{\text{inform, answer, justification}\}$, so $type \in \Psi_R \cup \Psi_I$. The set of all possible speech acts is denoted by $\Gamma = \text{percepts} = \text{actions}$. For each perception $p \in \text{percepts}$ an agent cycle results in a new epistemic state determined by $\mathcal{K}_D \circ_D p \circ_D \text{act}_D(\mathcal{K}_D \circ_D p)$. The set of all possible successive epistemic states of

agent \mathcal{D} is determined by the set of initial epistemic states $\Lambda_{\mathcal{D}}^0$ and all respective successor states for all possible perceptions and corresponding actions of \mathcal{D} . i. e. $\Omega_{\text{act}_{\mathcal{D}}, \circ_{\mathcal{D}}}(A_{\mathcal{D}}^0, \text{percepts}) = \{\mathcal{K} \mid \mathcal{K} = \mathcal{K}_0 \circ_{\mathcal{D}} p_0 \circ_{\mathcal{D}} \text{act}_{\mathcal{D}}(\mathcal{K}_0 \circ_{\mathcal{D}} p_0) \circ_{\mathcal{D}} \dots, p_0, \dots, p_i \subseteq \text{percepts}, i \in \mathbb{N}_0, \mathcal{K}_0 \in \Lambda_{\mathcal{D}}^0\}$. Our intuitive idea of secrecy preservation as given in Section 3 expresses that we want to assure that the secrecy preserving agent always maintains an epistemic state in which it believes that no other agent believes in something that it wants to keep secret. More exactly, it also distinguishes between secrets towards different agents and what it means to it that the information is kept secret. The term “*always maintains*” means that for all possible scenarios of communication the agent acts such that a safe epistemic state is maintained.

Definition 4 (Secrecy-preserving Agent). *Let $\mathcal{D} = (\mathcal{K}_{\mathcal{D}}, (\circ_{\mathcal{D}}, \text{act}_{\mathcal{D}}))$ be an agent and perceps a set of perceptions. An epistemic state $\mathcal{K}_{\mathcal{D}}$ is safe iff $\Phi \notin \text{Bel}(V_{\mathcal{A}}(\mathcal{K}_{\mathcal{D}}))$ for all $(\Phi, \text{Bel}, \mathcal{A}) \in \mathcal{S}(\mathcal{K}_{\mathcal{D}})$.*

Let $\Lambda_{\mathcal{D}}^0$ be a set of initial safe epistemic states. We call \mathcal{D} secrecy preserving with respect to $\Lambda_{\mathcal{D}}^0$ and perceps if and only if for all $\mathcal{K}_{\mathcal{D}} \in \Omega_{\text{act}, \circ}(\Lambda_{\mathcal{D}}^0, \text{percepts})$ it holds that $\mathcal{K}_{\mathcal{D}}$ is safe.

Example 6. We continue the previous example and check whether the initial \mathcal{K}_{emp} is safe. The set of answersets of P_{view} is $AS(P_{\text{view}}) = \{\{\text{attend_work}\}\}$. Consequently $\text{attend_scm} \notin \text{Bel}_{\text{skp}}^{\text{asp}}(P_{\text{view}})$ and \mathcal{K}_{emp} is safe.

Observation 2. *The definition of a secrecy preserving agent in Definition 4 satisfies (P4).*

We just defined the notion of a secrecy preserving agent. However, as discussed in Section 3 the actual resulting properties of secrecy preservation result from the properties of the change operation \circ and the attacker modeling. Moreover, the actual preservation of secrecy is realized by the means-end-reasoning of the agent. We elaborate these aspects in the next sections.

5 Belief Change and Secrecy

We decompose the belief change of an epistemic state into sub-operations on its components. Based on these we motivate and define properties with respect to secrecy preservation which formalize and concretize the ideas given in (P1) and (P5). Finally we give concrete instances of such operators for the ASP instance and show that they satisfy the defined properties.

5.1 Structure and Properties of the Change Operator

The change operator updates epistemic state of an agent upon incoming perceptions and actions. Formally, $\mathcal{K} \circ \tau = \mathcal{K}' = \langle \mathcal{B}', \Delta', \mathcal{I}' \rangle$. The change operator can be structured into several sub-operations for the different components of the epistemic state. Hereby the belief component is the only one being directly influenced by the new information, then the change of the desires is only dependent

on the changed beliefs and the update of the intentions on the changed beliefs and desires. Formally the sub-operations are $\circ_B : \mathcal{B} \times \Gamma \rightarrow \mathcal{B}$, $\circ_\Delta : \Delta \times \mathcal{B} \rightarrow \Delta$ and $\circ_{\mathcal{I}} : \mathcal{I} \times \mathcal{B} \times \Delta \rightarrow \mathcal{I}$. The update operations can then be represented as

$$\langle \mathcal{B}, \Delta, \mathcal{I} \rangle \circ \tau = \langle \mathcal{B} \circ_B \tau, \Delta \circ_\Delta (\mathcal{B} \circ_B \tau), \circ_{\mathcal{I}}(\mathcal{I}, \mathcal{B} \circ_B \tau, \Delta \circ_\Delta (\mathcal{B} \circ_B \tau)) \rangle.$$

In this section we focus on the belief change operation and its relevance to secrecy and elaborate the desire and intention change in the context of means-end-reasoning later on. The input speech act τ for the \circ_B operation can be either a perception or an action. In both cases it might be an informative or a requesting speech acts. All four cases have different semantics and lead to different changes. That is, the input has to be interpreted and represented in the language of the respective belief component we introduce translation operators $t_W : \Gamma \rightarrow \mathcal{L}_{At}^{asp}$ for the world view and $t_V : \mathfrak{A} \times \Gamma \rightarrow \mathcal{L}_{At}^{asp}$ for agent views. The result of the translation is then used to update the respective component by use of an *inner revision operator* $* : \mathcal{L}_{At}^{asp} \rightarrow \mathcal{L}_{At}^{asp}$. Secrets are updated on the basis of the agent's updated beliefs and views such that the change operator for secrets $*_{\mathcal{S}}$ is dependent on these as well as on the incoming information, i. e. $*_{\mathcal{S}} : 2^{\mathcal{L}_{\mathcal{S}}} \times \mathcal{L}_{At}^{asp} \times 2^{\mathcal{L}_{At}^{asp}} \times \Gamma \rightarrow \mathcal{L}_{\mathcal{S}}$. We define the changes of the \circ_B operator to the components by suboperations, such that

$$(V_W, \mathcal{V}, \mathcal{S}) \circ_B \tau = \langle V_W * t_W(\tau), \mathcal{V} * t_V(\mathcal{A}, \tau), *_{\mathcal{S}}(\mathcal{S}, V_W * t_W(\tau), \mathcal{V} * t_V(\mathcal{A}, \tau), \tau) \rangle (*)$$

We define a set of properties on the just defined operations which formalize the properties (S3), (P1) and (P5). For secrecy preservation it is necessary that the agent does not give up any secrets upon reflecting its own actions since it would be able to perform arbitrary actions without violating secrecy by abandoning its secrets. Thus, the agent must not be able to preserve a safe epistemic state by modifying its secrets.

Secrets-Invariance $_{\circ_B}$ If $\tau \in \text{actions}$ then $\mathcal{S}(\mathcal{B} \circ_B \tau) = \mathcal{S}(\mathcal{B})$

The *Secrets-Invariance* property is restricted to inputs that are actions. These actions are those of the agent itself and perceptions reflect changes in the environment or actions of other agents. For the latter the postulate should not hold. That is, an agent should not be able to ignore the fact that a secret has been revealed due to changes in the environment or actions of other agents. This is expressed in the following property.

Acknowledgment $_{\circ_B}$ If $\tau \in \text{percepts}$ then $\mathcal{B} \circ_B \tau$ is safe.

The changes to the set of secrets in order to achieve a safe epistemic state should be minimal. That is, a secret should not be weakened without a reason, i. e. it is violated, it should not be strengthened and it should be weakened minimally.

Min-Secrecy-Weakening $_{\circ_B}$ If $(\Phi, Bel, \mathcal{A}) \in \mathcal{S}(\mathcal{B})$ and $(\Phi, Bel', \mathcal{A}) \in \mathcal{S}(\mathcal{B} \circ_B \tau)$ with $Bel \neq Bel'$ then $\Phi \in Bel(V_{\mathcal{A}}(\mathcal{K}_{\mathcal{D}} \circ_B \tau))$ and there is no Bel'' such that $\Phi \notin Bel''(V_{\mathcal{A}}(\mathcal{K}_{\mathcal{D}} \circ_B \tau))$ with $s(Bel'') > s(Bel')$.

Another secrecy relevant property of belief change arises from the changes to views of other agents. An agent should not be able to preserve secrecy by ignoring the effects of its own actions on the beliefs of potentially attacking agents. In particular the information for some agent \mathcal{A} contained in an action of \mathcal{D} should be incorporated into \mathcal{D} 's view on \mathcal{A} . This is formulated by the next property.

Awareness_{o_B} If $\tau \in \text{actions}$ then $t_W(\tau) \in V_W(\mathcal{B} \circ_{\mathcal{B}} \tau)$ and for each $\mathcal{A} \in \mathfrak{A}$
 $t_V(\mathcal{A}, \tau) \in V_{\mathcal{A}}(\mathcal{B} \circ_{\mathcal{B}} \tau)$.

There might very well be actions which are not visible to all agents and therefore should also not affect the view on all agents.

Example 7. If agent \mathcal{D} is communicating privately with some agent \mathcal{A} it should change its view on \mathcal{A} but not its view on other agents.

This is achieved by use of appropriate translation operators which select the relevant information for each agent. For agents that are not affected by the information the transformation function returns the empty set.

Observation 3. *The satisfaction of Secrets-Invariance_o, Acknowledgment_o, Min-Secrecy-Weakening_o and Awareness_o of the belief change operator of an agent corresponds to the satisfaction of (P1), (P5) and (S3).*

We consider all of the properties defined in this section essential for a secrecy preserving agent, hence the goal is to define appropriate operators.

5.2 Concrete Revision Operations

In the following we define an instance of the translation operators t_V, t_W , the inner revision operator $*$ and the revision of secrets operator $*_{\mathcal{S}}$.

The translation operator, in accordance with (P1), has to consider information on two levels, on the one hand the actual information, that is the informational content of the speech act. On the other hand the meta-information about the speech act that has been performed which includes especially the information about the sender and the type of speech act and information revealed by these parameters. Both aspects have to be represented in the language of a logic program. We introduce an auxiliary logic program to support the representation of information on both levels for the ASP translation operators and to enable reasoning possibilities on the meta-information which are used for attacker modeling. To this end we reify literals, for each atom $A \in \text{At}$ introduce three constant symbols $C(A) = \{a, na, \lambda a\}$ the first two to represent the literals $A, \neg A$; λa can stand for both occurrences a and na . We define $const(L) = a$ if $L = A$ and $const(L) = na$ if $L = \neg A$. And $var(L) = \lambda a$ if $L = A$ or $L = \neg A$. We assume that the time is represented by a simple counter t and a literal $time(t)$, counting the agent cycles. The program P_{aux} consists of the following set of rules:

For all $A \in \text{At}$:

A	\leftarrow	$holds(a)$.
$\neg A$	\leftarrow	$holds(na)$.
$related(\lambda a, a)$.		$related(\lambda a, na)$.
$at(t)$	\leftarrow	$time(t), not\ time(s), s > t$.

The predicate $related(x, y)$ expresses that x is semantically related to y . We make use of the auxiliary construction to represent the informational content of a speech act and formulate the translation function as follows.

Definition 5 (Translation Function). *Let be $\mathcal{D} \in \mathfrak{A}, \tau = \langle A_s, \{A_{r_1}, \dots, A_{r_n}\}, type, L \rangle$ and counter = t . The translation functions of \mathcal{D} are defined as:*

$$t_V(\tau) = \begin{cases} \{type(A_s, const(L), t), L.\} & \text{if } \tau \in \Psi_I \\ \{type(A_s, var(L), t).\} & \text{if } \tau \in \Psi_R \end{cases}$$

$$t_W(\tau) = \begin{cases} \{type(A_s, const(L), t), L.\} & \text{if } A_s \neq \mathcal{D} \text{ and } \tau \in \Psi_I \\ \{type(A_s, var(L), t).\} & \text{else} \end{cases}$$

In general the information of a speech act is represented by the predicate $type(A_s, const(L), t)$ with the semantics that a speech act of type $type$ has been performed by agent A_S with logical content $const(L)$ at time t . For requesting speech acts $var(L)$ is used to represent that information related to L has been requested. Besides this representation of the information about the speech act the logical content has to be represented. For informational speech acts this is the actual literal L of which the agent has been informed. Hence L is added to the input set for the inner revision operator for informational speech acts, unless \mathcal{D} is updating its world view by its own action. The result of the translation operator is the input for the inner revision operator. As intended we revise a logic program by another one and face a standard belief revision problem and can make use of operators for it. For our ASP instance any operator satisfying the basic set of properties can be used. In particular *Success*: $Q \subseteq P * Q$, *Inclusion*: $P * Q \subseteq P \cup Q$ and *Vacuity*: If $P \cup Q$ is consistent, then $P \cup Q \subseteq P * Q$ are satisfied. For details refer to, e. g., [6].

As specified by the properties given above secrets are updated only by information about actions of other agent. In this case the secrets shall be modified minimally in order to preserve secrecy. To this end, we determine the strongest wrt. secrecy, that is the belief operator by which some information Φ is preserved in the current view \mathcal{V} . Formally: $curr(\Xi, \mathcal{V}, \Phi) = \arg \max s(Bel), Bel \in \{Bel \in \Xi \mid \Phi \notin Bel(\mathcal{V})\}$. Then we can define the change operator for secrets $*_S$ as

$$\mathcal{S}(\mathcal{K}) *_S (V'_W, \mathcal{V}', \tau) = \begin{cases} \mathcal{S}(\mathcal{K}) & \text{if } A_S = \mathcal{D} \\ \omega(\mathcal{S}(\mathcal{K}), \mathcal{V}') & \text{else} \end{cases}$$

with $\omega(\mathcal{S}(\mathcal{K}), \mathcal{V}') = \{(\Phi, Bel', \mathcal{A}) \mid (\Phi, Bel, \mathcal{A}) \in \mathcal{S}(\mathcal{K}) \text{ and}$

$$Bel' = \begin{cases} curr(\Xi, \mathcal{V}_A, \Phi) & \text{if } curr(\Xi, \mathcal{V}_A, \Phi) < Bel \\ Bel & \text{else.} \end{cases}$$

If the updating information is an action of \mathcal{D} , no changes are performed. Otherwise the belief operator of any secret whose assigned operator is stronger than the currently strongest one preserving secrecy is replaced by the latter. This means that only those secrets are modified which would be violated otherwise. We can show that this specification \circ_B satisfies the properties postulated previously.

Proposition 1. *Let \mathcal{D} be an agent, t_V, t_W and $*_S$ be operators as defined in this section and $P_{aux} \subseteq V_A(\mathcal{K}_D)$ and $P_{aux} \subseteq V_W(\mathcal{K}_D)$. Let $*_B$ be an ASP base-revision*

operator. The \circ_B operator of \mathcal{D} defined by as in (*) satisfies $\text{Secrets-Invariance}_\circ$, $\text{Acknowledgment}_\circ$, $\text{Min-Secrecy-Weakening}_\circ$ and Awareness_\circ .

Proof. Sketch: The satisfaction of $\text{Secrets-Invariance}_\circ$, $\text{Acknowledgment}_\circ$ and $\text{Min-Secrecy-Weakening}_\circ$ follow from the definition of $*_S$, the satisfaction of Awareness_\circ follows from Definition 5 and the satisfaction of the Success postulate by $*$.

6 Attacker Modelling

The principles P2 (simulation) and P3 (meta-inferences) of secrecy preservation laid out in Section 3 raise the need for adequate modeling of the background information and reasoning methods and capabilities of the attacker. Both over- and underestimating the capabilities of an attacker can lead to violation of secrecy. Hence modeling these is essential for realistic preservation of secrecy. In particular information about the declaration of secrets and meta-inference from \mathcal{D} 's behavior have to be considered. Which behavior is conspicuous and will lead to a violation of secrecy is heavily dependent on the reasoning capabilities and properties of the attacker. We define three properties of an attacker for secrecy preservation which \mathcal{D} might take into consideration.

Secret Aware \mathcal{A} knows which information \mathcal{D} does not want to reveal to \mathcal{A} if \mathcal{D} would believe it to be true.

Contradiction Sensitive \mathcal{A} considers self-contradictions of \mathcal{D} with respect to information it wants to keep secret as reason to infer the secret.

Refusal Sensitive \mathcal{A} considers the \mathcal{D} 's refusal to answer with respect to information it wants to keep secret as reason to infer the secret.

We present a simple version of an ASP approach to realize views of attackers satisfying the properties. To this end we consider the following set of rules which is then used to define programs which represent a specific property and can be modularly added to a view on an \mathcal{A} .

- (1) For each $(L, Bel, \mathcal{A}) \in \mathcal{S}(\mathcal{K}_{\mathcal{D}})$: $has_secret(D, const(L))$.
- (2) For all $A \in At$:
 $contradiction(D, \lambda a) \leftarrow inform(D, a), inform(D, na)$.
- (3) $holds(x) \leftarrow has_secret(D, x), contradiction(D, y), related(y, x)$.
- (4) $refused(D, x) \leftarrow request(_, D, x, t_1), not\ answer(D, y, t_2), at(t_2), related(x, y), t_2 = t_1 + 1$.
- (5) $holds(x) \leftarrow has_secret(D, x), refused(D, y), related(y, x)$.

Line (1) represents the information about the secrets \mathcal{D} has with respect to \mathcal{A} . In (2) it is expressed that \mathcal{A} infers that \mathcal{D} contradicted itself with respect to an atom A if it said both A and $\neg A$. If \mathcal{D} contradicted itself with respect to some secret information, then \mathcal{A} infers that the secret holds (3). Line (4) represents that \mathcal{A} infers that \mathcal{D} refused to answer about x if it was requested to do so and did not inform after the request. According to (5) \mathcal{A} infers that a secret holds if \mathcal{D} refused to answer with respect to it. We can define programs from the defined rules and formalize the properties given above in the ASP setting.

Definition 6. Let $P_{S\text{-aware}}^{\text{meta}} = (1)$, $P_{C\text{-sensitive}}^{\text{meta}} = P_{S\text{-aware}}^{\text{meta}} \cup (2) \cup (3)$ and $P_{R\text{-sensitive}}^{\text{meta}} = P_{S\text{-aware}}^{\text{meta}} \cup (4) \cup (5)$. An attacker modeling $V_A(\mathcal{K}_D)$ is secrecy aware if $P_{S\text{-aware}}^{\text{meta}} \subseteq V_A(\mathcal{K}_D)$, it is contradiction sensitive if $P_{C\text{-sensitive}}^{\text{meta}} \subseteq V_A(\mathcal{K}_D)$ and it is refusal sensitive if $P_{R\text{-sensitive}}^{\text{meta}} \subseteq V_A(\mathcal{K}_D)$.

Observation 4. The satisfaction of the properties contradiction sensitive and refusal sensitive corresponds to the properties (P3) (a) and (b), respectively.

The determination of contradictions inflicted by \mathcal{D} and the one of refusals can and should be more elaborate and can easily be extended and formulated by more complex logic programs, but are outside of the scope here.

7 Means-End-Reasoning and Secrecy Preservation

We equipped the agent with the abilities to be aware of its secrets and to detect violation of secrecy. The question now is what are the necessary properties on the desire and intention change operators for secrecy preservation.

In any BDI system desire and intention change operators are implemented in one way or the other which is. Here we give a general model of intention change and show the relevant properties for secrecy preservation. Any agent has to determine how it can satisfy its intentions, high-level intentions are resolved down to atomic intentions $\text{AtInt} \subseteq \mathfrak{I}$ which can be satisfied by a single action $\alpha(I)$. In any case at some point the options to satisfy some intention have to be evaluated and one of the options has to be chosen. The options for a given intention are determined by the options function $\text{options} : \mathcal{I} \rightarrow 2^{\mathcal{I}'}$ and the evaluation results in a preference relation $<$ on the possible options. Here we assume that all intentions can directly be resolved to an atomic intention and set $\mathcal{I}' = \text{AtInt}$. Then the set of maximally preferred options from which one is selected by the agent is set to $\text{pref}(\text{options}(I)) = \max_{<}(\text{options}(I))$.

To show that an agent is secrecy preserving, as given by Definition 4, we have to show that it prefers secrecy preserving actions over non-secrecy preserving ones and that it always has a secrecy preserving option.

We generalize the change operator \circ to intentions as input. In general this allows to determine the effects of the satisfaction of arbitrary intentions. For our presentation here we can set $\mathcal{K} \circ I = \mathcal{K} \circ \alpha(I)$. Given an epistemic state \mathcal{K} and an intention I , an option $o \in \text{options}(I)$ is *safe* iff $\mathcal{K} \circ o$ is safe. Based on this definition we define a secrecy relevant property on the preference relation on options.

Confidentiality-preference For all $I \in \mathcal{I}$, for $o, o' \in \text{options}(I)$ if o is safe and o' is not then $o > o'$.

In combination with an options function we can show the agent choses secrecy preserving options, if they exist.

Lemma 1. If $<$ satisfies confidentiality-preference, then if there is some safe option $o \in \text{options}(I)$, then for all $o' \in \text{pref}(\text{options}(I))$, o' is safe.

Proof. The lemma follows directly from the definitions of *pref* and confidentiality-preference.

A preference relation satisfying *confidentiality-preference* alone is not sufficient to guarantee secrecy preservation since it is dependent on the existence of a safe option. Hence we define the following property of an options function.

Existence For all $I \in \mathcal{I}$ and safe \mathcal{K} there exists $o \in \text{options}(I) : \mathcal{K} \circ o$ is safe.

We can now formulate the dependency of the notion of a secrecy preserving agent, Definition 4, and the properties of option selection as defined in this section.

Proposition 2. *Let $>$ be a preference relation on AtInt and options an options function of an agent \mathcal{D} . If $>$ satisfies Confidentiality-preference and options satisfies Existence, then \mathcal{D} is secrecy preserving.*

Proof. Sketch: The proposition follows from Lemma 1 and the definitions of Confidentiality-preference, Existence, safe options and safe epistemic state.

8 Related Work and Conclusion

We presented an theoretical, conceptional and practical account of secrecy from the subjective view of an autonomous epistemic agent. We formulated properties of secrecy and secrecy preservation and developed a framework for an ASP-based instance satisfying them. We have shown in [7] that other many aspects of notions of secrecy such as [1] and [4] can be captured by our underlying model.

To the best of our knowledge no subjective account of agent based secrecy nor a concrete model or implementation of a secrecy preserving agent system has been presented so far. The closest to this is the preliminary account of integrating techniques for controlled query evaluation for databases [1] into an agent system, as presented in [2]. The database techniques are naturally limited to a fixed client-server architecture and to query-answer scenarios. In [2] it is proposed to use a *sensor* to check the agents actions prior to execution and modifying them if necessary similar to the approaches from database theory for a negotiation scenario. The actual realization of this approach is left open. We argue that instead of adding an controlling instance secrecy has to be integrated into the agents' reasoning, deliberation and means-end reasoning processes to achieve autonomous secrecy preserving agents apt to perform well in a dynamic setting. However, secrecy is a very special epistemic goal which calls an appropriate epistemic model, operators and actions as presented in this work and can hardly be captured by the few existing approaches for maintenance goals, e. g. [5].

We see our model and implementation as a good basis for the further theoretical investigation as well as the implementation of secrecy preserving agents. It opens a plethora of possibilities for further investigation. In current work we run empirical evaluations and integrate advanced deliberation [9] and means-end reasoning techniques [8] in our model and implementation, and investigate further properties of secrecy in this model and the relation to other approaches.

Acknowledgements. This work has been supported by the DFG, Collaborative Research Center SFB876, Project A5. (<http://sfb876.tu-dortmund.de>)

References

1. Biskup, J.: Usability confinement of server reactions: Maintaining inference-proof client views by controlled interaction execution. In: Kikuchi, S., Sachdeva, S., Bhalla, S. (eds.) DNIS 2010. LNCS, vol. 5999, pp. 80–106. Springer, Heidelberg (2010)
2. Biskup, J., Kern-Isberner, G., Thimm, M.: Towards enforcement of confidentiality in agent interactions. In: Proc. of the 12th Intl. Workshop on Non-Monotonic Reasoning (NMR 2008), pp. 104–112 (2008)
3. Gelfond, M., Leone, N.: Logic programming and knowledge representation: the A-Prolog perspective. *Artificial Intelligence* 138 (2002)
4. Halpern, J.Y., O’Neill, K.R.: Secrecy in multiagent systems. *ACM Transactions on Information and System Security* 12, 5:1–5:47 (2008)
5. Hindriks, K.V., van Riemsdijk, M.B.: Satisfying maintenance goals. In: Baldoni, M., Son, T.C., van Riemsdijk, M.B., Winikoff, M. (eds.) DALT 2007. LNCS (LNAI), vol. 4897, pp. 86–103. Springer, Heidelberg (2008)
6. Krümpelmann, P., Kern-Isberner, G.: Belief base change operations for answer set programming. In: del Cerro, L.F., Herzig, A., Mengin, J. (eds.) JELIA 2012. LNCS, vol. 7519, pp. 294–306. Springer, Heidelberg (2012)
7. Krümpelmann, P., Kern-Isberner, G.: On agent-based epistemic secrecy. In: Proc. of the 14th Int’l Workshop on Non-Monotonic Reasoning, NMR 2012 (2012)
8. Krümpelmann, P., Thimm, M.: A logic programming framework for reasoning about know-how. In: Proc. of the 13th Int’l Workshop on Non-Monotonic Reasoning (NMR 2010) (2010)
9. Krümpelmann, P., Thimm, M., Kern-Isberner, G., Fritsch, R.: Motivating agents in unreliable environments: A computational model. In: Klügl, F., Ossowski, S. (eds.) MATES 2011. LNCS, vol. 6973, pp. 65–76. Springer, Heidelberg (2011)
10. Rao, A.S., Georgeff, M.P.: BDI-agents: from theory to practice. In: Proc. of the 1st Int’l Conference on Multiagent Systems (ICMAS 2005) (1995)
11. van der Torre, L.: Logics for Security and Privacy. In: Cuppens-Bouahia, N., Cuppens, F., Garcia-Alfaro, J. (eds.) DBSec 2012. LNCS, vol. 7371, pp. 1–7. Springer, Heidelberg (2012)

Bid-Price Control for the Formation of Multiagent Organisations

Marc Premm¹, Tobias Widmer¹, and Paul Karänke²

¹ University of Hohenheim, Department of Information Systems 2,
Stuttgart, Germany

{marc.premm,tobias.widmer}@uni-hohenheim.de

² FZI Research Center for Information Technology, Karlsruhe, Germany
karaenke@fzi.de

Abstract. Agents that participate in a multiagent organisation must receive a reasonable compensation for delivering services to this organisation. Otherwise, the agents would refrain from joining the organisation due to their self-interest. Thus, the formation of multiagent organisations is no mechanical process, but subject to considerations of the involved agents. We approach this decision problem by a bid-price approach from quantity-based Revenue Management to maximise each individual agent's expected revenue. The proposed method is evaluated in a simulation with competing service provider agents. The results suggest that our approach is robust for most cases with low demand and outweighs non-discriminating formation processes when supply exceeds demand.

1 Introduction

The cooperation between autonomous agents is a key concern in multiagent systems research. Agents provide services to other agents, but as agents are self-interested, they expect something in return as compensation. Beside the question how agents should cooperate to solve a common or individual task, an agent that identifies a task of interest has to decide in the first place with whom to cooperate. The set of agents that decided and agreed to cooperate is denoted as a multiagent organisation (MAO).

For the formation of MAO, potential members have to decide about their participation. For a provider agent, its participation is beneficial if the compensation covers at least the cost for providing the requested service. For a consumer agent, its participation is beneficial if its valuation of the received service is equal or greater than the price charged. This formation problem has been subject of much work in multiagent systems research. Current approaches in the area of multiagent coalitions [1], however, either assume complete information [2] or agents that are not concerned with their personal payoffs [3]. The literature for MAO formation, existing approaches mainly focuses on the definition of constructs and models, but does not address the decision problem in detail [4,5,6].

We approach this decision problem from a Revenue Management (RM) perspective. RM summarises various methods that maximise expected revenues

through price discrimination. Our proposal enhances MAO formation by differentiating prices for two or more consumer agents. RM has its origin in the airline industry and has also successfully been applied in the hotel and car rental industries [7]. Findings from the operations research literature suggest that RM is well-suited for situations, in which resources have a fixed capacity, customers can be segmented in different groups and the product is not storable [8]. Multiagent systems also show these characteristics. Hence, we argue to adopt RM to multiagent systems for the maximisation of each agent’s expected revenue.

We consider a scenario where agents provide services to other agents. For example, agent a_1 represents an actor that has access to a knowledge base. When the agent joins a MAO, it is committed to deliver the service on request and therefore receives a compensation for this service. The revenue maximisation can be achieved by offering the same service with varying non-functional parameters, e.g., quality of service, to extend the potential service consumer set. There are several possible discrimination criteria: The quality of service can vary in the guaranteed response time or in the level of detail of the delivered information. Furthermore, the provider agent may be capable of providing a service that is hosted in a data centre. In this case there are all discrimination criteria possible that are usually included in a service level agreement. Examples may include availability, cancellation restrictions, and off-peak (e.g. night-time) bookings. Each provider agent can also take the role of a consumer agent, if it has appropriate contracts with other service provider agents. Agents can form new MAOs (figure 1) or join existing ones (figure 2).

However, to merely offer different service levels is not sufficient due to the risk that most of the resources could be used to provide less profitable services. RM provides means to protect higher priced services in favour of lower priced services to maximise expected revenues. Thus, the objectives of this research are (1) to develop a mechanism that maximises expected revenues through price discrimination for the formation of MAOs and (2) to demonstrate the efficacy of the proposed artefact through a simulation experiment. The contribution is a novel method for agents to decide whether the participation in a particular MAO increases their expected revenues.

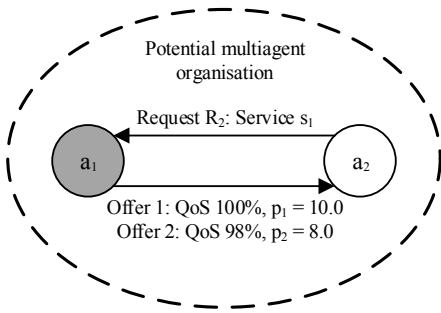


Fig. 1. Formation of new MAO

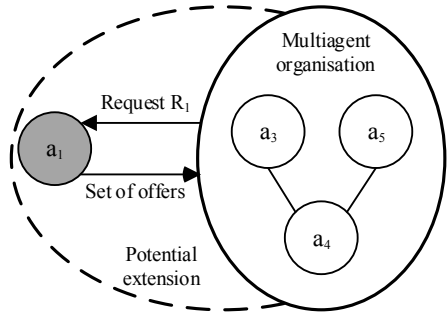


Fig. 2. Join of existing MAO

The remainder of this paper is structured as follows. Section 2 gives an overview to the theoretical background. In section 3, we present a formal framework that enables price discrimination in MAO formation. The bid-price control approach is presented in section 4. Section 5 reports the evaluation. Section 6 is the conclusion.

2 Theoretical Background

2.1 Formation of Multiagent Organisations

Multiagent systems (MAS) are shared by multiple agents and multiagent environments provide communication means to enable agent interaction. Further, MAS are typically open without a central designer (i.e., there are *multiple loci of control*) [9,10].

The emerging organisational context between agents defines the agents' relationship with each other [11]. Agent organisations provide a framework of constraints and expectations about the agents' behaviour with focus on decision making and action of specific agents [12]. Horling and Lesser present an overview over various multiagent organisational paradigms and describe methods used for the organisation formation processes [13].

Wooldridge et al. define a MAO as a collection of roles that stand in relationships to each other and take part in institutionalised interaction patterns with other roles [5]. Ferber et al. characterise MAOs as follows: (i) multiple agents that manifest a behaviour, (ii) can be partitioned into sub-organisations (groups) that may overlap, and (iii) agents have roles which relate their behaviour to the organisation activity [6].

The notion of *coalitions* constitutes a prominent example of organisations within multiagent systems [1]. The concepts used in designing coalition formation are mainly drawn from cooperative game theory and employ automated negotiation among self-interested agents to form coalitions that are stable with regards to some appropriate metric. Coalition formation approaches are closely related to the formation of MAOs. Generally, coalition formation comprises the formation of agent coalitions in situations of partial conflict of interest, though complete information is traditionally assumed [2]. Other approaches relax several assumptions but consider coalition formation where agents cooperate that are not concerned with their personal payoffs, i.e., without any conflict of interest [3]. In contrast, in MAOs, members are bound by contracts, have incomplete information, and do not necessarily have same compatible goals.

The concept of *virtual organisations* has been applied to leverage the formation of organisations within multiagent systems. Barbuceanu and Fox employ an agent system to analyse different entities in an inter-organisational supply chain [14]. An explicit usage of agent technology as metaphor and software paradigm for the design and the operation of a virtual organisation is done by Fischer et al. [4]. In particular, the authors present different possibilities how software agents can contribute to the formation of virtual organisations and highlight several auction mechanisms that can be used for virtual organisation formation.

Partner selection and mutual agreements with regards to the structure of the virtual organisation are realised through agent negotiation processes. A formal framework consisting of autonomous agents which act as rational decision makers within virtual organisations was developed by Zarour et al. [15]. Agents take roles in the virtual organisation to foster the association of obligations and interdictions opposed by the virtual organisation. The proposed model incorporates negotiation operations that are used for the distribution and achievement of sub-goals in view of the leading agent's main goal.

Hence, existing approaches to MAO formation mainly focus on auctions and automated negotiation among agents but do not consider price discrimination for the formation process.

2.2 Revenue Management

Much progress has been made on developing methods for maximising expected revenues by price differentiation. Since revenue and utility maximisation are closely related in multiagent organisations, we review contributions from the RM literature that may be beneficial for the formation of multiagent organisations.

The term RM – also known as Yield Management – summarises different approaches that maximise expected revenues, e.g. price differentiation by offering the product or service in different booking classes or the overbooking of available resources supposing that not all of the resources will be actually demanded.

RM has its origin in the American airline industry after the deregulation of the market in 1978. Today, these RM methods are applied in various industries, whose products or services have three main characteristics in common: The product or service is perishable, it has a fixed capacity, and its potential customers can be segmented according to their price-sensitivity. Weatherford and Bodily [8] outline these common characteristics and proposes a comprehensive taxonomy. McGill and van Ryzin [16] as well as Chiang et al. [7] give a brief overview over existing approaches and over other industries in which RM might be applied.

The applied methods can be distinguished into quantity-based and price-based RM [17]. Price-based RM tries to adapt the price for a service to increase revenue by applying auctions or dynamic pricing strategies. Dynamic pricing approaches are widely spread, especially in non-traditional RM industries, where Bitran and Calentey [18] give an overview. However, traditional RM industries like airlines, hotels or car rental companies mainly rely on quantity-based RM. Approaches in quantity-based RM include single-resource capacity control, where only one single resource is considered, the extension for multiple products, known as network capacity control, as well as overbooking, where more resources are sold than actually available. Kultti shows the practical equivalence of auctions and posted pricing, so we focus on posted pricing with quantity-based RM used by traditional RM industries [19]. From the perspective of RM, auction-based approaches as discussed in 2.1 [3,4,15] could also be regarded as RM approaches.

To the best of our knowledge, methods from quantity-based RM have not been applied to MAO formation. Hence, for designing a method for MAO formation, we draw from literature on network capacity control. The reason is that the

formation problem incurs multiple services; therefore, single resource capacity control is not feasible.

3 Formal Framework

This section provides a formal framework for the formation of MAOs where potential participants offer their services in multiple booking classes.

3.1 Overview

We consider autonomous software agents within a multiagent system that have the ability to deliver certain services. To deliver a service, the agents have to agree upon a contract that includes obligations for both participants. With this contract, the agents form a MAO that obliges one participant to deliver the service and the other one to pay a compensation. The provided services are assumed to have marginal costs near zero, such that every additionally sold service increases total utility for the agent. Compensation payments for a binding participation in a MAO are posted publicly. This posting allows every potential participant to choose the appropriate booking class while the offerer has to find proper discrimination attributes to hinder service consumers to select a lower fare than their willingness to pay would allow. It is assumed that agents know one or more discrimination criteria that allow to differentiate certain groups of potential participants with different willingness to pay. In many cases this discrimination can be achieved by applying different quality of service settings for each booking class (e.g., response time, availability, etc.). The ability to assert discrimination requires a certain market power of the actor, i.e., a mono- or oligopoly market situation [17].

Services are not storeable and each time an agent has resources left that can not be used otherwise, it suffers opportunity costs. However, to apply quantity-based RM approaches, the services need to be reservable. That is, the contribution of an agent may be necessary for a fixed time span in the future. This might be the case if the contribution is only needed once or the contribution can help the MAO to handle periodic peaks.

The variable t_r indicates the time for the arrival of requests with $t_r \in \{0, \dots, T_r\}$ while t_p is used for the planning horizon with $t_p \in \{0, \dots, T_p\}$. These two time scales are related by the function $f_T(t_p) = t_r$. It is assumed that at most one request per time period t_r arrives, however, with the second time horizon for t_p , this restriction does not affect the planning of service provisioning.

3.2 Services and Resources

The agents' contributions to the MAOs are the provision of certain services. N_S denotes the amount of available services in the multiagent system. Let $S = \{s_1, \dots, s_{N_S}\}$ be the set of all services. These services are offered by the agents in N_B different booking classes: Let $B = \{b_1, \dots, b_{N_B}\}$ be the set of booking classes

with decreasing quality of service, i.e., the contract for any service s_i will have lower quality attributes in booking class b_m than in b_n with $m > n$. The provision of a service is compensated by the MAO by paying the agreed compensation, which is hereinafter referred to as the price for the corresponding service. Let \mathbf{p}^b be the price vector for every booking class $b \in B$ with $\mathbf{p}^b = (p_1^b, \dots, p_k^b, \dots, p_{N_S}^b)^\top$ and prices p_k^b for service $s_k \in S$.

Every agent participating in a MAO needs resources to provide services to the MAO. Let N_J be the number of resources. The available resources at time period t_r will be represented by the $N_J \times T_p$ matrix X_{t_r} . Assuming that all resources are unused in $t_r = 0$, the total capacity is described by X_0 . For every time period of the request time horizon the resource capacity left is expressed as follows:

$$X_{t_r} = \left(x_{j,t_p}^{t_r} \right) = \begin{pmatrix} x_{1,1}^{t_r} & \cdots & x_{1,T_p}^{t_r} \\ \vdots & \ddots & \vdots \\ x_{N_J,1}^{t_r} & \cdots & x_{N_J,T_p}^{t_r} \end{pmatrix}$$

As already mentioned in the example, each service consumes a certain amount of resources. Under the assumption that the resource consumption does not vary over time and is fix for every booking class $b \in B$, we define the $N_S \times N_J$ matrix M_b that maps every service s to the consumed resources j :

$$M_b = (m_{s,j}^b) = \begin{pmatrix} m_{1,1}^b & \cdots & m_{1,N_J}^b \\ \vdots & \ddots & \vdots \\ m_{N_S,1}^b & \cdots & m_{N_S,N_J}^b \end{pmatrix}$$

Each element $m_{s,j}$ of the matrix relates to the number of resources j that are used with the delivery of one instance of service s . Hence, n services will consume $n \cdot m_{s,j}$ units of resource j .

3.3 Demand and Requests

We assume that each agent will face at most one request in every time period t_r for the participation in a MAO. A request always refers to the offered services and not to the resources as the usage of the latter may vary from agent to agent. Analogous to the resource matrix X_{t_p} , the request in t_r is also represented by a $N_S \times T_p$ matrix $R_{t_r} = (r_{s,t_p})$ for the amount of requested services s for every period of the planning horizon t_p . The agent receiving the request offers its participation in different booking classes and the requesting agent has to decide which booking class to choose, respectively whether it wants the offering agent to participate at all.

The relation between price and the expected demand will be represented by the price-quantity function $f(\cdot)$. Most literature in Revenue Management assumes that the price discrimination enforced by additional restrictions does not effect the price-quantity function itself. However, the introduction of additional

restriction to services like lower quality of service commitments or cancellation restrictions automatically lower the value of the offered service for the interested agents. Hence, we apply an extended model according to [20], which allows to take these degradation costs into account. The price-quantity function is also dependent on the booking class b : $f_{s,b,t_p}(p_s^b)$ with $f_{s,b,t_p}(p_s^b) \geq f_{s,b+1,t_p}(p_s^{b+1}) \forall b < N_b$. These degradation costs lead to the overall request quantity $Q_{s,b}$ for service s in booking class b and time period t_p [20]:

$$Q_{s,b,t_p} = \underbrace{f_{s,b,t_p}(p_s^b)}_{\text{demand for class } b} - \underbrace{\sum_{k < b} Q_{s,k,t_p}}_{\text{demand for higher classes}}. \quad (1)$$

4 Bid-Price Control

This section presents an approach for agents to decide whether the participation in a particular multiagent organisation increases its expected revenues. The model introduced in section 3 is similar to other network-capacity problems in RM. Auctions are not applicable, due to the fact that the offering agents are facing an oligopoly market and in every time period t_r there is at most one request.

4.1 Decision Model

We assume that the compensation an agents receives for participating in a MAO is fixed for every service and booking class for the relevant time horizon. However, the agents have to decide for every booking class, whether the participation in the requested MAO with the compensations and restrictions of this specific booking class would be beneficial for the agent or not. To enable the agent to answer a participation request in a timely manner, we use a bid-price control, which calculates so called bid-prices before the actual request arrives. This approach is widely used in traditional RM industries for network-capacity problems. The bid-price is sort of a reservation price for the offering agent that represents a lower limit, under which a participation is expected to have a suboptimal outcome. Although bid-prices controls are not always optimal, [21] show that they are asymptotically optimal with the right bid-prices. The prices themselves for each service and booking class have to be determined in a preceding step, which is not part of this paper. We assume that this step of maximising expected returns by specifying prices and discrimination levels per booking class has been already done and the corresponding values are given.

The availability control for booking classes with bid-prices is a common solution for many traditional RM problems. One advantage of this approach is that the decision whether the participation in the potential MAO is beneficial for the agent can be calculated for a fixed amount of resources in linear time dependent on the affected time period. Even though we assume that prices are fixed for the planning horizon, we are able to implement varying prices by adding a booking class with the same restrictions but a different price. If more than one booking class is

available, the requesting agent would always choose the cheaper alternative and by closing this booking class the price for the service automatically rises.

As the resource consumption of the execution of each service may vary with the chosen booking class, the bid-prices are based on the resources and calculated for every combination $x_{j,t_p}^{t_r}$ of resources j and time period t_p of the planning horizon. The available resources left and the bid-prices are both dependent on t_r . Hence, the $N_J \times T_p$ matrix Π_{t_r} of bid-prices is structured similarly to X_{t_r} :

$$\Pi_{t_r} = \left(\pi_{j,t_p}^{t_r} \right) = \begin{pmatrix} \pi_{1,1}^{t_r} & \cdots & \pi_{1,T_p}^{t_r} \\ \vdots & \ddots & \vdots \\ \pi_{N_J,1}^{t_r} & \cdots & \pi_{N_J,T_p}^{t_r} \end{pmatrix}$$

Let $p(R_{t_r}, b)$ be the total price for booking class b for an incoming request at t_r . With the request matrix R_{t_r} and the fixed price vector \mathbf{p}^b the price can be calculated with a matrix multiplication: $p(R_{t_r}, b) = \left(\mathbf{p}^b{}^\top R_{t_r} \right) \mathbf{a}_{T_p}$ with the T_p -dimensional vector $\mathbf{a}_{T_p} = (1, 1, \dots, 1)^\top$. The resource consumption of request R_{t_r} is represented by $M_b^\top R_{t_r}$.

On the basis of these variables, the agent can decide whether the participation in a specific booking class is beneficial, and thus, whether this booking class will be offered. Therefore, the agent has to compare the compensation it will receive for the participation in this MAO with its reservation price. The reservation price for a bundle of resources respectively for multiple time periods is defined as the sum over all affected bid-prices, each multiplied with the requested amount. Every booking class b that will be offered for an request R_{t_r} with resource consumption $M_b^\top R_{t_r} = (\theta_{i,j})$ and price $p(R_{t_r}, b)$ has to fulfill the following inequation:

$$\sum_{j \in [1, N_J]} \sum_{t_p \in [1, T_p]} \pi_{j,t_p} \theta_{j,t_p} \leq p(R_{t_r}, b) \quad (2)$$

Based on this rule, the agents can decide, which booking classes to offer for an incoming request. The available resources in the next time span $t_r + 1$ if request R_{t_r} will be offered and subsequently booked by the requesting agent in booking class b is $X_{t_r+1} = X_{t_r} - M_b^\top R_{t_r}$.

4.2 Calculation of Bid-Prices

The main challenge of the bid-price approach is to calculate the bid-prices themselves. An intuitive solution is to use the expected return of the available resources left at time t_r to calculate the opportunity costs. If the corresponding time span t_p is reached and resources are not used in any MAO, these idle resources would be wasted. Let $V_{t_r}(X_{t_r})$ be the expected return for the available resources left X_{t_r} at t_r . With the assumption that V_{t_r} has a derivative $\frac{\partial}{\partial x_{j,t_p}} V_{t_r+1}(X_{t_r})$ for resource j and time span t_p , the bid-prices can be calculated respectively approximated as follows [17]:

$$\pi_{k,l} := \frac{\partial}{\partial x_k} V_{t_r+1}(X_{t_r}) \approx V_{t_r+1}(X_{t_r}) - V_{t_r+1}(X_{t_r} - A_{k,l}) \quad (3)$$

where $A_{k,l} = a_{m,n}$ is a $N_J \times T_p$ matrix with

$$a_{m,n} = \begin{cases} 1 & m = k, n = l \\ 0 & \text{otherwise} \end{cases}$$

Let q_{s,b,t_p} be the optimal amount of provided service s in booking class b in time period t_p . We can calculate V_{t_r} with the quantity constraints for the expected demand Q_{s,b,t_p} :

$$\begin{aligned} V_{t_r}(X_{t_r}) &= \max_{q_{s,b,t_p} \forall s,b,t_p} \sum_{t_p=f_T^{-1}(t_r)}^{T_p} \sum_{s \in S} \sum_{b \in B} p_s^b q_{s,b,t_p} \\ &= \sum_{t_p=f_T^{-1}(t_r)}^{T_p} \max_{q_{s,b,t_p} \forall s,b,t_p} \sum_{s \in S} \sum_{b \in B} p_s^b q_{s,b,t_p} \end{aligned} \quad (4)$$

$$\begin{aligned} \text{s.t.} \quad & q_{s,b,t_p} \leq Q_{s,b,t_p} \\ & \sum_{s \in S} \sum_{b \in B} q_{s,b,t_p} m_{s,j}^b \leq x_{j,t_p}^{t_r} \quad \forall j, \forall t_p \\ & q_{s,b,t_p} \in \mathbb{N} \end{aligned}$$

Since the only difference between $X_{t_r} = (x_{j,t_p}^{t_r})$ and $X_{t_r} - A_{k,l} = (\hat{x}_{j,t_p}^{t_r})$ in equation 3 is at $t_p = k$ and q_{s,b,t_p} is independent in t_p , we infer $x_{j,t_p}^{t_r} = \hat{x}_{j,t_p}^{t_r} \quad \forall t_p \in \{f_T^{-1}(t_r), \dots, T_p\} \setminus \{k\}$. Hence, the calculation of $\pi_{t_p,j}$ can be simplified to

$$\pi_{t_p,j} \approx \max_{q_{s,b,t_p} \forall s,b,t_p} \sum_{s \in S} \sum_{b \in B} p_s^b q_{s,b,t_p} - \max_{\hat{q}_{s,b,t_p} \forall s,b,t_p} \sum_{s \in S} \sum_{b \in B} p_s^b \hat{q}_{s,b,t_p} \quad (5)$$

$$\begin{aligned} \text{s.t.} \quad & q_{s,b,t_p}, \hat{q}_{s,b,t_p} \leq Q_{s,b,t_p} \\ & \sum_{s \in S} \sum_{b \in B} q_{s,b,t_p} m_{s,j}^b \leq x_{j,t_p}^{t_r} \quad \forall j \\ & \sum_{s \in S} \sum_{b \in B} \hat{q}_{s,b,t_p} m_{s,j}^b \leq x_{j,t_p}^{t_r} - 1 \quad \forall j \\ & q_{s,b,t_p}, \hat{q}_{s,b,t_p} \in \mathbb{N} \end{aligned}$$

Equation 5 enables each agent to calculate the bid-price matrix, and thus, decide which booking classes will be offered or not.

5 Evaluation

We evaluate the proposed method analytically on the computational complexity and describe an experimental evaluation afterwards. Finally, we present and discuss the results of the experiments.

5.1 Analytical Evaluation

The analytical evaluation focuses on the computational complexity of the bid-price calculation. We therefore formulate the following theorem:

Theorem 1. *The computation problem for the expected valuation is NP-complete.*

Proof. For NP-membership, each solution for the expected valuation can be obviously verified in polynomial time. For NP-hardness, the problem is reduced to the NP-hard knapsack problem [22].

The knapsack problem is defined as follows. Let U be a finite set and $u \in U$. Further, $g(u)$ denotes the weight and $b(u)$ the valuation of element u , and K is a positive integer. Further, $a(u)$ is a non-negative integer for each $u \in U$. Now maximize $\sum_{u \in U} a(u)b(u)$ s.t. $\sum_{u \in U} a(u)g(u) \leq K$ (cf. [23]). Any instance of the knapsack problem can be reduced to our optimization problem as follows. Let $t_p = 1$, i.e., we consider one single time period only, and let $|S| = 1$ with $s \in S$ fixed, i.e., s is the only service available in S , then we can rewrite equation (4) as

$$V_{t_r}(X_{t_r}) = \max_{q_{s,b,1}} \sum_{b \in B} p_s^b q_{s,b,1} \tag{6}$$

with constraint $\sum_{b \in B} q_{s,b,1} m_{s,j}^b \leq x_{j,1}^{t_r}$ for all j . Further, set $B = U$, $q_{s,b,1} = a(u)$, $p_s^b = b(u)$, $m_{s,j}^b = g(u)$ for all $u \in U$, and $x_{j,1}^{t_r} = K$. Obviously, this reduction can be done in polynomial time. Hence, the computation problem for the expected valuation is NP-complete.

The calculation of the expected valuation is NP-complete, but the optimisation is done separately for each time period and its complexity rises with the number of offered services and booking classes. However, the presented bid-price approach provides the advantage that this computationally expensive task can be pre-calculated such that an incoming request is answered in a timely manner, even for a large number of services and booking classes.

5.2 Experimental Evaluation

The objective of our experiment is to evaluate the proposed bid-price approach with price discrimination in comparison to conventional posted pricing approach without different booking classes. The problem for the provider agent is to decide whether he will join the organisation for a lower compensation or not. It runs the risk of using the scarce resources for underpaid services.

Setup. For the simulation of the multiagent system, we use Repast Symphony [24]. The maximisation operations for the calculation of bid-prices (equation 5) are performed by CPLEX. Consumer agents send service requests to the provider agents and receive corresponding offers. Consumer agents choose the cheapest

service offer that meets their restrictions, and choose randomly if they get more than one similar offer. Provider agents try to maximise their individual revenue as marginal costs are assumed to be near than zero. The time horizon is set to $T = 300$ with $t_r = f_T(t_p) = t_p$. To minimise the influence of single parameter settings, we only consider one type of service which requires one type of resource for every booking class what implies $m_{s,j}^b = 1 \forall s, j, b$.

On the service provider side, one regular provider offers only in the highest booking class and therefore will always return exactly one offer, provided that resources are available for the requested time period. The other provider agent uses the proposed method to control the availability of different booking classes. In this simple case, there is only one additional booking class that has lower quality of service and therefore a lower price. The aim of the provider agent is to get additional revenue from consumer agents that have a lower reservation price than the original target group. The price for the highest booking class is for both types of provider agents set to 10.0 monetary units (MU), while the bid-price provider offers the second booking class for 8.0 MU. The expected demand is calculated using history data and provider agents predict requests for all upcoming time periods based on the average data of the last 20 time periods.

Requests are generated by two types of consumer agents: C_1 has a lower price sensitivity and accepts offers in the most expensive booking class $b = 0$ only, and C_2 has a higher price sensitivity and accepts offers in both booking classes. These consumer agents generate requests for every time step and send them to every service provider. The reservation prices are normally distributed for both consumer agents: $X_c \sim \mathcal{N}(\mu_c; \sigma_c^2)$. For consumer agent C_1 we set $\mu_{C_1} = 12.0$ and $\sigma_{C_1} = 4.0$, respectively for consumer agent C_2 : $\mu_{C_2} = 7.0$ and $\sigma_{C_2} = 3.0$.

Each request also has some time parameters that are generated randomly. The starting time in relation to the current time period and duration of the service request follow a Poisson distribution with $\lambda = 30.0$ respectively $\lambda = 20.0$ for C_2 , and $\lambda = 5.0$ respectively $\lambda = 10.0$ for C_1 . Hence, C_1 , which is willing to pay more for the service, will request services in a shorter time in advance. How many services will be requested per time period follows a normal distribution. However, the parameters μ and σ are randomly generated for each request and follow a Poisson distribution with $\lambda_\mu = 30.0$ and $\lambda_\sigma = 2.0$ for C_1 respectively $\lambda_\mu = 15.0$ and $\lambda_\sigma = 2.0$ for C_2 . Consequently, the average demand intensity is the same for both consumer agent types.

Results and Discussion. The simulation is performed 100 times each for different supply/demand ratios with one bid-price-based and one regular provider on a competing market. Figure 3 shows the results, where demand has been constant in all scenarios, but the available resource capacity of each provider has been varied. To remove transient effects we measured the gained revenue and resource utilisation that affect time periods 50 to 250 for means and standard deviations.

Figure 3(a) shows the average revenue per tick gained by both providing agents dependent on the their total capacity. If demand for high priced services is higher than supply, the discrimination has no positive effect for the

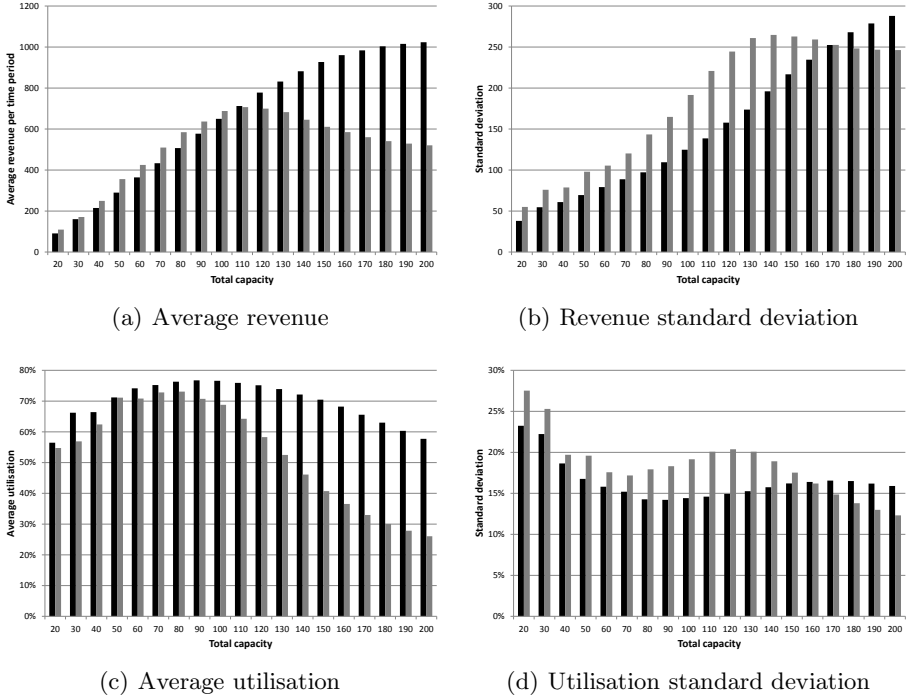


Fig. 3. Average revenue and resource utilisation per time period for bid-price provider (black) and regular provider (grey)

revenue of the providing agent. The consumer agents with a lower reservation price tend to book earlier in advance, consequently the bid-price approach should protect all available capacity for consumers with a higher reservation price and can not perform better than the regular provider. The real advantage of the proposed method is visible when the available resources are expanded. The bid-price provider is then able to serve additional consumers and even poach consumers that also accept lower-priced booking classes from the regular provider. Even though the bid-price provider gained slightly lower revenue in some constellations, figure 3(b) shows that the volatility of the revenue per time period is lower for the bid-price approach, and does not exceed the regular provider until average revenues are significantly higher. There is also a slight advantage for the bid-price approach when resources are very scarce, as there are more potential consumers and therefore a higher chance that resources left can be allocated.

Besides the gained revenue, we also measured the resource utilisation of each service provider. Figure 3(c) shows that the bid-price approach achieved a higher utilisation than the regular one in all scenarios. This even holds for the standard deviation of the utilisation, so the utilisation is higher and steadier. Again, not until a significant utilisation surplus over the regular provider is reached, the volatility of resource utilisation also increases for the bid-price provider.

6 Conclusion

This work presents a bid-price approach for the MAO formation using quantity-based Revenue Management methods. Therefore, we provide a formal framework that allows agents to provide their services in different booking classes to extend the set of potential consumers. Although the calculation of bid-prices is \mathcal{NP} -complete, the bid-prices can be precalculated, so every request can be answered in a timely fashion. The experimental evaluation with artificial data suggests that the proposed approach is robust in most scenarios with low demand and outweighs conventional approaches when supply exceeds demand. These findings imply a potential increase of revenues for agents that provide services and have idle resources. The proposed method introduces different booking classes and thus expands the set of potential consumers for each provider agent.

Our approach is not limited to a specific domain, since price discrimination can be applied in different contexts. However, the method is limited to requests with fixed time spans. Future research could extend the method to allow requests with uncertain duration and sliding time horizon. In addition, the calculation of bid-prices mainly relies on the prediction of future demand, hence, extended forecasting methods [25] promise to gain additional revenues. Since service requests and thus the service consumers do not reveal the preferred booking class, this prediction process needs additional input variables. Findings from RM research indicate that overbooking positively affects revenue [7].

Acknowledgement. This work has been supported by (1) the eHealthMonitor project (<http://www.ehealthmonitor.eu>), funded by the European Commission under contract FP7-287509, and (2) the project MIGRATE!, funded by the German Federal Ministry of Economics and Technology (BMW, FKZ 01ME11052).

References

1. Sandholm, T., Lesser, V.: Coalitions among computationally bounded agents. *Artificial Intelligence* 94, 99–137 (1997)
2. Kahan, J.P., Rapoport, A.: *Theories of coalition formation*. L. Erlbaum Associates (1984)
3. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. *Artificial Intelligence* 101(1-2), 165–200 (1998)
4. Fischer, K., Müller, J., Heimig, I., Scheer, A.: Intelligent agents in virtual enterprises. In: *Proceedings 1st International Conference on the Practical Application of Intelligent Agents and Multi Agents Technology* (1996)
5. Wooldridge, M., Jennings, N.R., Kinny, D.: The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems* 3(3), 285–312 (2000)
6. Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: An organizational view of multi-agent systems. In: Giorgini, P., Müller, J.P., Odell, J.J. (eds.) *AOSE 2003*. LNCS, vol. 2935, pp. 214–230. Springer, Heidelberg (2004)

7. Chiang, W.C., Chen, J., Xu, X.: An overview of research on revenue management: current issues and future research. *International Journal of Revenue Management* 1(1), 97–128 (2007)
8. Weatherford, L.R., Bodily, S.E.: A taxonomy and research overview of perishable-asset revenue management: Yield management, overbooking, and pricing. *Operations Research* 40(4), 831–844 (1992)
9. Huhns, M.N., Stephens, L.M.: Multiagent systems and societies of agents. In: Weiss, G. (ed.) *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pp. 79–120. MIT Press (1999)
10. Jennings, N.: On agent-based software engineering. *Artificial Intelligence* 117, 277–296 (2000)
11. Ferber, J., Gutknecht, O.: A meta-model for the analysis and design of organizations in multiagent systems. In: *Proceedings of the Third International Conference on Multi-Agent Systems, ICMAS 1998* (1998)
12. Bond, A.H., Gasser, L. (eds.): *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers, San Mateo (1988)
13. Horling, B., Lesser, V.: A survey of multiagent organizational paradigms. Technical report, University of Massachusetts (2004)
14. Barbuceanu, M., Fox, M.S.: The information agent: an infrastructure agent supporting collaborative enterprise architectures. In: *Third Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises* (1994)
15. Zarour, N., Boufaïda, M., Seinturier, L.: Supporting virtual enterprise systems using agent coordination. *Knowledge and Information Systems* 8(3), 330–349 (2005)
16. McGill, J.I., van Ryzin, G.J.: Revenue management: Research overview and prospects. *Transportation Science* 33(2), 233–256 (1999)
17. Talluri, K.T., van Ryzin, G.J.: *The Theory and Practice of Revenue Management*. Springer, New York (2004)
18. Bitran, G., Calentey, R.: An overview of pricing models for revenue management. *Manufacturing & Service Operations Management* 5(3), 203–229 (2003)
19. Kultti, K.: Equivalence of auctions and posted prices. *Games and Economic Behavior* 27, 106–113 (1999)
20. Botimer, T., Belobaba, P.: Airline pricing and fare product differentiation: A new theoretical framework. *Journal of Operational Research Society* 50, 1085–1097 (1999)
21. Talluri, K., van Ryzin, G.: An analysis of bid-price controls for network revenue management. *Management Science* 44(11), 1577–1593 (1998)
22. Karp, R.: Reducibility among combinatorial problems. In: Miller, R., Thatcher, J. (eds.) *Complexity of Computer Computations*, Plenum Press, New York (1972)
23. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York (1979)
24. North, M., Collier, N., Ozik, J., Tatara, E., Macal, C., Bragen, M., Sydelko, P.: Complex adaptive systems modeling with repast simphony. *Complex Adaptive Systems Modeling* 1(1), 3 (2013)
25. Syntetos, A.A., Boylan, J.E., Disney, S.M.: Forecasting for inventory planning: A 50-year review. *The Journal of the Operational Research Society* 60, 149–160 (2009)

Peer-to-Peer Network for Flexible Service Sharing and Discovery

Andrii Zhygmanovskyi and Norihiko Yoshida

Department of Computer Science, Saitama University, Japan
{andrew,yoshida}@ss.ics.saitama-u.ac.jp

Abstract. In this paper we present P2P-based approach for storing, sharing and discovering services, which can be utilized as a base for agent cooperation in distributed multiagent system. The approach is loosely based on Peer-to-Peer based Web Service Discovery (PWSD) architecture and can be seen as its extension and enhancement. Unlike most other P2P-based approaches it allows flexible search queries since all of them are executed against internal database present at each overlay node. The infrastructure proposed in the paper can serve as a base for wide range of applications including distributed scheduling scenarios and service mash-ups.

Keywords: peer-to-peer, agent cooperation protocol, service discovery, service sharing.

1 Introduction

Nowadays computing is becoming more and more dominated by distributed applications which, in turn, grow to be more complex and require sophisticated protocols for communication and cooperation between participants. On the other hand, many distributed application scenarios can be executed by a set of intelligent agents thus minimizing human participation or eliminating it at all. Since one of the main characteristics of multiagent systems is cooperation, there have already been proposed numerous agent cooperation protocols, such as Contract Net [1] or Recruiting Interaction Protocol [2], which describe an interaction between agents toward finding the most suitable agent for performing given task. However, classical approaches for agent cooperation are foremost designed from the viewpoint of agent capabilities and rarely take the distributed nature of multiagent systems into proper consideration. More specifically, some cooperation models such as teamwork model [3] or joint-intentions model [4] assume there are some shared memory locations that agents can use to share their mental state and goals, but this is difficult to assume for distributed environments. Other models like the Group Situation based Cooperation model [5] assume that there are some global coordinator entities which can act as control entities for other agents but it tends to make the model prone to errors and create bottlenecks in a large-scale environment. These issues show a necessity for researching new means of building distributed agent cooperation environment.

In the current paper we propose an approach which is based on agent capabilities, that is we associate each agent with set of services it provides, so that multiagent system can be seen as distributed network of service providers. Dating back to the days when service-oriented computing was only gaining its importance, it was stated that every system which contains interacting services must ensure that the following fundamental processes are implemented in a correct and efficient way. First, a service must be described usually using structures like keywords or property-value pairs. Second, a service must declare its bindings and interfaces, as well as other information that allows clients to invoke it. Third, there must be a mechanism which allows for publishing all that information making a service discoverable. And lastly, service discovering facilities must exist, namely, query construction, routing and execution, as well as results propagation back to the requesting party. Again, since multiagent systems have inherently distributed nature of, it is natural that managing the services which are provided by each agent should be done in a decentralized way. Peer-to-peer (P2P) based approach has already established a good reputation for storing and managing content in a decentralized way, so it seems appropriate to use it for the distributed service discovery. That is, nodes in the P2P overlay network, where each node stores a description of some service in the network, act not only as routing providers and data location services, but also as servers providing service access.

Despite the fact that P2P approach is successfully used for content storage and management for about 15 years, service management systems based on it are still quite sparse and, as a general rule, they either remain within academia or even don't evolve further than proof-of-concept stage. The earliest successful P2P-based service discovery networks include Hypercube[6] and Speed-R. Also, many of proposed solutions are based on Semantic Web approach for describing services. For instance, in the approach described in [7] services are described using DAML-S while service publishing and discovery mechanism based on JXTA technology. Similar approach is used in [8], but Gnutella P2P overlay is used instead of JXTA. In [9] authors propose to perform service search according to business aspect of services and not based on service descriptions itself. This approach aims to facilitate better service composition during service execution phase later. On the other hand, [10] presents a DHT-based P2P overlay service network with partial keywords and wildcards matching support. In this network authors use a special kind of DHT key mapping - locality preserving mapping based on Space Filling Curves.

In this paper we propose a Web Service discovery architecture based on structured P2P overlay network (namely, Chord [11]), which utilizes platform-independent attribute-value based method for describing high-level properties of web services and elaborate algorithm for service discovery that allows variety of queries including range and fuzzy ones. This system is loosely based on the PWSD (Peer-to-Peer based Web service discovery) architecture presented in [12] and can be seen as an extension and enhancement of this approach. PWSD system concerns web service discovery in a P2P environment with the implementation all of four fundamental processes mentioned above. It is a lightweight approach that uses attribute-value based service description without resorting to complex data description frameworks (i.e. Semantic Web). It is based on Chord DHT and by extending it only a little contributes greatly to

simplicity and modularity of the approach. The similarity of the goals and common points in the ways to reach them are the main reasons why exactly PWSD was chosen as the starting point of our solution.

Main differences between the approach taken in PWSD and the system proposed in this article can be summarized as follows. First, while NVTtree structure introduced in PWSD and (a,v)-graph structure used in this paper bear a lot of similarities, our approach do not use any explicit serialization format for the service descriptions (like XML-based WSDL format used in PWSD) and the algorithms for building abstract service description itself differ considerably. More detailed comparison between NVTtree and (a,v)-graph is provided later in this article. Second, the approach to service descriptions storing taken in this paper naturally allows range and fuzzy queries which is not possible in PWSD system. And finally, while PWSD architecture includes an extension of Chord overlay network (named XChord), we show that actually there is no need for such extension to be based on some particular overlay network and different structured P2P overlays could be used together or interchangeably.

While being inherently a framework that allows people collaborate on providing, discovering and using services, the approach presented in this paper is designed in a way that makes it possible to employ intelligent agents at some (or all) nodes. That is, according to one of the definitions, which states that multiagent system is a “system in which several interacting, intelligent agents pursue some set of goals or perform some set of tasks” [13], the network for service sharing and discovery can delegate some of its activities to an agent which resides in overlay node and is capable of performing such tasks as negotiation and distributed scheduling minimizing human user participation in them. In fact, we regard this research direction as the most promising way to expand our system to real applications.

The rest of the article is organized as follows. Section 2 presents the overview of PWSD network. Section 3 present the architecture of proposed service sharing and discovery network including service description, service storing and service discovery mechanisms. Section 4 presents the experimental results. Conclusions are given in the section 5.

2 Web Service Discovery in PWSD

Given that we have a valid service description, we obtain a set of keys from it and pass them to the hash function such as MD5 to generate a set of Hash IDs (HID) which are used to locate appropriate peers by means of Peer-to-Peer routing algorithms. Subsequently, HIDs are published to the target peers, which cache the description in router repositories thus completing the publishing process. The process of service locating is roughly the same. Thus, the key step in service publishing and locating process is looking up a peer node according to HID, which was made possible by extending routing algorithm of Chord DHT.

Each peer in PWSD acts as a service peer (SP), which not only provides Web service access, but also acts as a peer in the Peer-to-Peer overlay network. Several logical machines can share one piece of hardware as well as it's possible for a single logical machine to consist of several physical machines. Each logical machine consists of three active components (Web Service Discovery Interface component, core

component, router) and one passive component called local repository. The roles and structure of these components are described in detail in [12].

Service locating algorithm specifies how to route the requests to the service peers who satisfy service request conditions. In PWSD, the service request is expressed in XML, since it's consistent with XML-based service descriptions stored in the destination service peers. However, the routing algorithm, Chord, in underlying Peer-to-Peer overlay network only supports exact match queries within each service description keyword. That's why authors presented an extension to Chord algorithm called XChord which supports XML based conditional match. In PWSD, WSDL is used to describe the Web service interface, and the service description is generated based on the content of WSDL document and the description that user inputs before publishing.

Service description generation in PWSD is based on the structure called NVTree (node-value tree). More precisely, an XML based tree node extraction approach is used for generating service descriptions. Example service description and its corresponding node-value tree can be found in the original paper [12]. It is worth noticing that only significant elements in service description will be extracted and inserted into the NVTree, and only the meaningful nodes in the NVTree will be used to generate a hash value, which in turn will be used as a hash key for service description and will be inserted into P2P overlay network. In PWSD, simple node-splitting method is used to extract each node-value pair to form NVTree and independently map them onto a key. However, only the leaf nodes in NVTree have a pair of node and value. Furthermore, in order to preserve the hierarchical relationship, the parent node of the leaf node is also extracted. Those nodes whose values consist of several words are further divided into single word value based nodes. After splitting the NVTree into separated simple description nodes, they are concatenated as strings, which are, in turn, passed to the hash function to produce hash IDs. These hash IDs are used as keys to insert into the underlying P2P overlay using XChord algorithm.

To search for a Web service, the client specifies query conditions by composing an XML document. Also, conditions can be combined using logical operators (or, and) to form composite queries. The query is processed in a way similar to the service description processing. That is, the XML document is transformed to the NVTree and then to simple description nodes. These nodes are concatenated as strings and hash ID is obtained using the same hash function from the underlying P2P overlay network. Since those hash IDs are, in fact, the keys in the underlying P2P overlay, usual DHT-based search is performed to answer the query. Finally, search results are further combined according to the logical operators used (set union for logical and, set intersection for logical or).

3 Service Sharing and Discovery Network

3.1 Overview

The idea of building service sharing and discovery network based on P2P overlay itself is not innovative, since it allows for avoiding many problems that arise in centralized scenarios like single point of failure, poor scalability or lack of robustness.

Nevertheless, even nowadays most of P2P-based systems deal with simple content sharing, which is fundamentally different from functionality of sharing services. Still, there is a range of problems in common that are present in both cases, most crucial of them being appropriate descriptions of items shared (content or services) as well as creating flexible and efficient search functionality that provide results relevant to the users criteria as much as possible. PWSD system described in the previous part of this article was designed to provide a solution to both these problems and it does so to some extent. From the other hand, authors of that approach made several architectural decisions which actually make the system less flexible in terms of basic problems mentioned above, namely service descriptions and service discovery. In this article we try to eliminate the shortcomings of the PWSD system by introducing more elaborate and platform-independent approach for describing services and formulating search queries.

The pivotal point of the system proposed is original platform-independent format of service descriptions. It is similar to the separate node descriptions obtained from NVTree in PWSD, but more exactly, it is based on Intentional Name System naming approach described in detail in [14]. Unlike PWSD, where service descriptions are initially given in XML format, the approach in our system is based on abstract graph-based attribute-value description of services which are called (a,v)-graph. The underlying serialization format of the graph is actually not important since it is not the part of the framework itself. We also propose several ways for building description graph, including utilization of existing descriptions, automatic description building and manual description input.

Despite the fact that NVTree and (a,v)-graph have much in common, it is the differences that make (a,v)-graph structure more flexible. In order to get a better view on advantages of the approach proposed in this paper we need to outline those differences more clearly. As can be seen from the way of building the NVTree, it is basically a tree representation of a XML-based (actually, WSDL-based) service description which is further split into atomic attribute-value pairs, where value nodes contain typed data (in case type information is not available, the value is assumed to be of string type). Besides, the information present in NVTree comprises all data found in the original service description, including the routing, binding and service owner information about the service itself. On the other hand, while (a,v)-graph uses the same attribute-value structure, it serves as basic format for the service description itself, which, among other ways, could be obtained by transforming corresponding service descriptions, including XML-based ones. Besides, (a,v)-graph structure contains only information that is meaningful for the service discovery, and also can be easily extended to contain the value type specification, so search queries could be executed in a type-aware way. Also it is important to note that service owner and binding information are included in the (a,v)-graph as a special node which is essential to the further execution of discovered services.

Another difference concerns the point of how those attribute-value pairs are used in the hashing process. The approach in PWSD is as follows - attribute and value string

are concatenated and passed as an argument to the hash function which determines the place where this piece of information is stored in the overlay network. The approach in the (a,v)-graph is similar but instead of hashing attribute and value altogether, the hash is computed only for attribute node and corresponding subgraph of the (a,v)-graph is copied to the responsible node according to obtained hash value. This way the structure stored at the responsible is still an (a,v)-graph, which allows for the queries to be much more flexible, which includes fuzzy and range queries. Flexibility of the queries is also stipulated by the fact that actual mechanism of (a,v)-graph storage is not defined, so different implementations can choose the best one for the specific needs.

The next step for services discoverability after descriptions is the way how they are stored in a distributed manner in P2P overlay network. Similarly to PWSD we chose Chord overlay for this purpose, but while the authors of PWSD decided to extend Chord DHT algorithm, scheme for storing and discovering service descriptions proposed in this article is overlay-independent. That is, the algorithms of storing, removing, updating and locating the services are defined in the layer completely disconnected from the DHT layer and deal with only with abstract notions like “responsible node”. This way it is possible to have multiple layers of overlays for service storage (for instance, to increase reliability or distribute the load), and those overlays, in fact, do not have to be the same. This approach will be described in more detail in the next sections. Finally, we present abstract format of querying the distributed database of service descriptions which makes possible wide range of conditions including fuzzy conditions, range queries and complex values lookup.

Table 1. Key differences between NVTree and (a,v)-graph

NVTree	(a,v)-graph
<ul style="list-style-type: none"> • is a result of WSDL document transformation • there is no distinction between binding properties and properties of a service itself • hash value is based on concatenated attribute-value pair • responsible node is storing only a hash value like a piece of content • supports only exact match queries within given keyword 	<ul style="list-style-type: none"> • is basic abstract format for storing service properties • contains binding information as a special node • hash value is based solely on the attribute • a structure that responsible node is storing is also an (a,v)-graph • supports fuzzy and range queries

3.2 Service Description

Each service in the network is described using attribute-value format, forming so called (a,v)-pairs, where attribute stands for the arbitrary property of a service. While attributes can obviously be only of string type, values are not required to be of string type only, so they can be of any type which is, in principle, queryable, serializable and can be efficiently stored by the underlying DHT storage mechanism. So, for example, as it is shown on the Fig. 1, value can be an array (like ‘location’ attribute, which contains latitude and longitude values) or even an arbitrary object (like ‘availability’ attribute). One more significant difference from the approach taken in PWSO is that all (a,v)-pairs form a connected graph, called (a,v)-graph, which always includes one extra node, that represents routing and binding information for the service itself. In real life situations it’s not rare when one node provides access to several services, so each service is described using (a,v)-graph.

One of the most challenging issues for any new approach or framework is to provide compatibility with existing technologies, which are already widely used. It is important to note again that (a,v)-graph approach is just an abstract model of service description, therefore it should be seen as an output of some model transformation function. In our system the possible ways of obtaining the resulting (a,v)-graph model are as follows:

Manual Input. The simplest case where user enters all service description information manually, usually with some kind of GUI, but batch information input, for instance, by uploading the file with multiple attribute-value pairs is also possible. Regardless of how the data is put into the system, it always can be processed and transformed to the underlying (a,v)-graph serialization format

Transformation of Existing Service Description. To address the issue of compatibility the system must have a way to obtain (a,v)-based descriptions from existing ones. In our case this is achieved by applying necessary mode transformations, exact nature of which depends on the representation of existing models. But since in most cases existing services are described using XML-based industry standards like WSDL or OWL-S, Extensible Stylesheet Language Transformations (XSLT) language seems to be the most appropriate choice for model transformation in this case, due to its high expressive power and ability to output virtually any kind of data format using XML data as input. It is important to note that in most cases existing service description need to be transformed to (a,v)-based one only partly, since low-level details of a service (like protocol name, input parameters enumeration, IP address etc) are generally not searched upon. However, those low-level details can still be present in (a,v)-graph in the service description node to facilitate the process of binding and routing when the service is actually used.

Automatic Augmentation. Among service description properties there are often ones that are highly important as a search criteria but normally are neither supposed to be

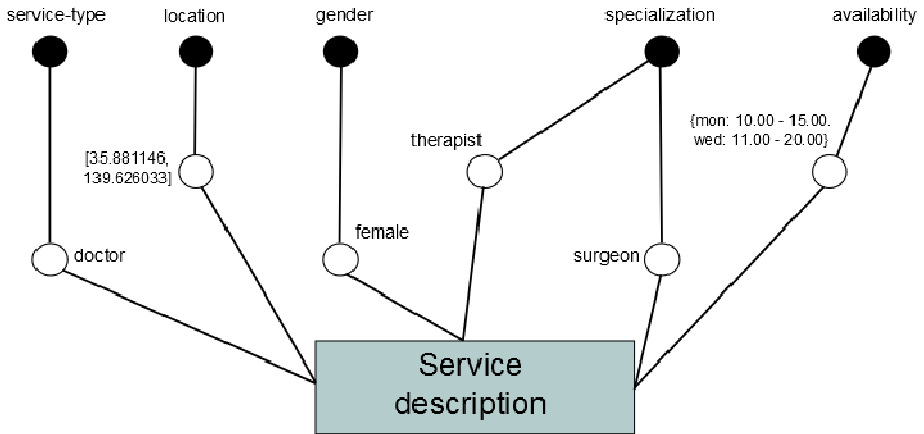


Fig. 1. (a,v)-graph for service description

input by humans nor present in traditional static service descriptions. Examples of such properties are current geographical location of the service, status of the job queue, various QoS data (like performance or latency) and sharing network data (like current node reputation). All those data can actually be obtained automatically using various means, including automatic location detection, internal monitoring functionality or network monitoring and QoS protocols.

3.3 Service Description Storing

In the approach, proposed in this article, Chord DHT peer-to-peer overlay is used to store service descriptions in a distributed manner. As usual, in order to store content in DHT we need to define what will act as a key and what exactly will be stored at nodes in the overlay. In our system, we apply hash function to each attribute name and decide the node in the overlay which is responsible for this attribute - in case of Chord DHT it is successor of a key. In terms of our approach, this node is called

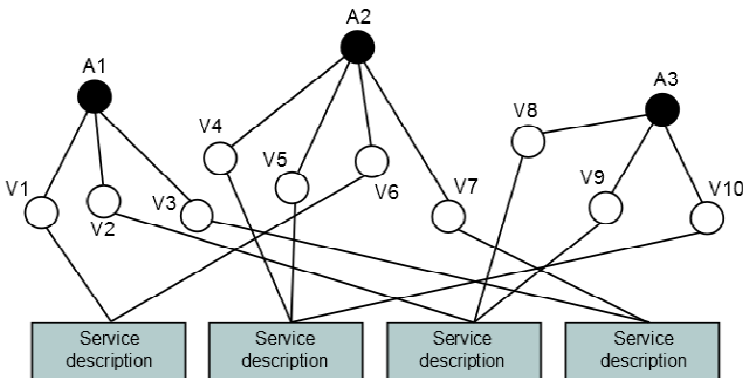


Fig. 2. Merged (a,v)-graph example

responsible for the attribute and therefore will store subgraphs of a form $[attribute, value, service\ description]$, that is, subgraphs based on given attribute, of all (a,v)-graphs in the network. In the result, we obtain a structure called merged (a,v)-graph in each node that is responsible at least for one attribute. Example of merged (a, v)-graph is shown in Fig. 2.

Next we present formalized version of service description storing algorithm. Each peer P in the overlay owns two graphs, namely, $P.OS$ – graph for the services it owns, and $P.SS$ – graph for the services from other peers it stores. The formal definition of both graphs is as follows: $P.OS = (V_V \cup V_A \cup \{sd\}, E_{A,V} \cup E_{V,SD})$ and $P.SS = (V_V \cup V_A \cup SD, E_{A,V} \cup E_{V,SD})$, where V_V – set of nodes that correspond to the attributes of the service, V_A – set of nodes that correspond to the values of the attributes, sd – node which contains the low-level service description (including an information for the owner node), SD – set of sd nodes, $E_{A,V}$ – set of edges $(v_A, v_V) | v_A \in V_A, v_V \in V_V$ and $E_{V,SD}$ – set of edges $(v_V, sd) | v_V \in V_V, sd \in SD \wedge sd = sd$. Then, we assume that for each peer in the overlay the following two functions are defined: h – hash function used to build an overlay, and $find$ – function that returns the node from the overlay by hash value. The pseudocode for service description storing algorithm is shown in Fig. 3.

It's not essential to the architecture of the system how exactly merged (a,v)-graphs are stored at the node, but storage mechanism should be chosen in a way which makes possible answering complex queries like comparison, substring checks, set operations and so on. Therefore, the most appropriate choices for storage mechanism are relational databases or document-oriented databases, both of which have their own advantages and disadvantages.

```

foreach service  $s$  from  $P.OS$ 
  foreach  $v_A \in s.V_A$ 
     $H := h(v_A)$ ;
     $P := find(H)$ ;
    if not exists  $v_A^* \in P.SS.V_A$  where  $v_A^* = v_A$ 
       $P.SS.V_A := P.SS.V_A \cup \{v_A\}$ ;
    end if
     $sd^* := get\ sd^* \text{ from } P.SS.SD \text{ where } sd^* = s.sd$ ;
    if  $sd^*$  is NULL
       $P.SS.SD := P.SS.SD \cup \{s.sd\}$ ;
       $sd^* := s.sd$ ;
    end if
     $V := \{v_V \in s.V_V | (v_A, v_V) \in s.E\}$ ;
    foreach  $v \in V$ 
       $P.SS.V_V := P.SS.V_V \cup \{v\}$ ;
       $P.SS.E := P.SS.E \cup \{(v_A, v)\}$ ;
       $P.SS.E := P.SS.E \cup \{(v, sd^*)\}$ ;
    end foreach
  end foreach
end foreach

```

Fig. 3. Service description storing algorithm

3.4 Service Discovery

Among one of the most significant drawbacks of DHT-based P2P overlays is that the principles of content storage and its association with a key usually allows only for exact query conditions when searching. Alternative, there are some elaborate approaches addressing this issue which, but again they usually offer only wildcard matching. The approach we propose in this article is based on the way service descriptions are stored in the overlay network, that is, using merged (a,v)-graph.

Firstly, we assume that each search query in the system is submitted in the form, shown in Fig. 4 or can be represented as such. Here we let $op_i(A^i)$ be some operators ($=, !=, \text{contains}, >, <$ etc) defined on the sequence of attributes A_i as parameters, A_i - sequence of attributes $[a_1, a_2, \dots, a_n]$ (where n is arity of the operator op_i) and lop_i denote logic operators *OR* or *AND*. This format itself is very generic, so we think that it represents most of meaningful search queries submitted in P2P networks. You can see a concrete example of the query in Fig. 5.

$$op_1(A_1) \quad lop_1 \quad op_2(A_2) \quad lop_2 \quad \dots \quad lop_{n-1} \quad op_n(A_n)$$

Fig. 4. Generic search query format

```

service-type = 'doctor'
AND
(location = 'Tokyo' OR location = 'Yokohama')
AND
availability > 'June 15th, 2012, 2:00PM'
AND
availability <= 'June 17th, 2012, 9:30AM'

```

Fig. 5. Example of search query

Algorithm of search query routing and execution is formalized below. Note that the algorithm actually doesn't require any extensions for the underlying DHT algorithm. The query issued by the peer in the form shown in Fig. 4 can be represented as a graph $QU = (Q \cup LOP, E)$, where $Q = \{q_i = (a_i, op_i) | i = 1 \dots n\}$ - the set of query terms, $LOP = \{lop_i | i = 1 \dots n\}$ - the set of logical operators and $E = \bigcup_{i=1}^n \{(q_i, lop_i)\} \cup \{(lop_i, q_{i+1})\}$. In addition to peer functions *h* and *find* introduced in the section 3.3, we assume that each peer have function *evaluate* which returns the result of evaluating a query term again internal database of the peer. Similarly, we define graph $RES = (R \cup LOP, E)$, where $R = \{r_i | i = 1 \dots m\}$ - initially empty set of results obtained from peers after evaluating a query term, LOP is the same as $QU.LOP$ and $E = \bigcup_{i=1}^n \{(r_i, lop_i)\} \cup \{(lop_i, r_{i+1})\}$. Also we define the set of pairs $C = \{(OR, \cap), (AND, \cup)\}$ which shows the one-to-one correspondence between logical operators and set-theoretical operations. Pseudocode for the algorithm is shown in Fig. 6.

```

foreach  $q \in QU.Q$ 
   $H := \mathbf{h}(q.a)$ ;
   $P := \mathbf{find}(H)$ ;
   $R := P.\mathbf{evaluate}(q)$ ;
   $RES.R := RES.R \cup \{R\}$  for corresponding  $lop \in LOP$ ;
end foreach
return  $r_1 C[lop_1] r_2 C[lop_2] \dots C[lop_{n-1}] r_n$ 
  where  $r_i \in RES.R, lop_i \in RES.LOP$ ;

```

Fig. 6. Search query routing and execution algorithm

4 Results

The implementation of the framework described above has been done using Chord overlay and consists of the following main modules: Chord overlay, the service storage, statistics and operational information collection module and the visualizer. To demonstrate the system operation we chose a domain which consists of three types of entities, namely hotels, museums and baseball matches, which act like services to be stored and queried in the P2P-based services network. The attributes (and data types of respective values) used to describe each type of service are shown in the table 2.

Given services are then put into the service storage network according to the procedure described in the section 3.3, that is, a hashing function defined in the underlying Chord overlay is applied to each attribute and according to its value the attribute

Table 2. Service types and attributes for the service P2P network example

Hotel	Museum	Baseball match
<ul style="list-style-type: none"> • name • suite type (single or double) • price for the one night stay • meals (yes/no field) • number of available rooms • location 	<ul style="list-style-type: none"> • name • type of exhibition (painting, sculpture, photograph) • entrance fee • start date of exhibition • end date of exhibition • time the exhibition opens (daily) • time the exhibition closes (daily) • location 	<ul style="list-style-type: none"> • league • (Central, Pacific) • competing teams (pair) • venue name • price • date of the match • time the match starts • time the match ends • venue location

for the given service is stored at the according node. The storage layer itself is implemented using MongoDB, document-oriented “no-SQL” database engine, which suits best for the storage of attribute-value data due to its flexibility and relative simplicity. In the course of the implementation of the storage layer it became clear that at least three database tables per node are needed for the full coverage of basic data manipulation functions, that is (1) putting service to the storage, (2) updating service stored at the overlay, (3) removing service from the storage and (4) getting service from the storage based on the query. The tables and their structure are shown in the table 3.

Table 3. Database tables used for storing services at storage node and their structure

Local service data	Remote service data	Service attributes data
comprehensive data about given node services locally	attribute data for services in the overlay network; all attributes the node is responsible of are stored	information about responsible nodes for attributes of given service; service, responsible for storing this information, is determined by hashing service name itself;

After the data is placed in the overlay we can perform services search using the query format described earlier in this article. The sample query and result returned for it are shown in Fig. 7.

```

Waiting until the overlay stabilizes...
Populating the data storage...
Ready
get location like "Ginza" and price < "10000":int32
*****
RESULTS:

=====
Service name: Hotel_430907
name = Hotel Gracery Ginza <Responsible node: 111 = 133.38.238.161:16912, 1>
suite-type = single <Responsible node: 40 = 133.38.238.161:19926, 1>
price = 5000 <Responsible node: 40 = 133.38.238.161:19926, 1>
meals = False <Responsible node: 105 = 133.38.238.161:19051, 1>
location = Ginza 7-10-11, Chuo-ku, Tokyo Prefecture <Responsible node: 105 = 133.38.238.161:19051, 1>
rooms-available = 0 <Responsible node: 166 = 133.38.238.161:17157, 1>
Service owner: 111 = 133.38.238.161:16912, 1
=====

=====
Service name: Hotel_655822
name = Hotel Gracery Ginza <Responsible node: 111 = 133.38.238.161:16912, 1>
suite-type = single <Responsible node: 40 = 133.38.238.161:19926, 1>
price = 6000 <Responsible node: 40 = 133.38.238.161:19926, 1>
meals = True <Responsible node: 105 = 133.38.238.161:19051, 1>
location = Ginza 7-10-11, Chuo-ku, Tokyo Prefecture <Responsible node: 105 = 133.38.238.161:19051, 1>
rooms-available = 6 <Responsible node: 166 = 133.38.238.161:17157, 1>
Service owner: 111 = 133.38.238.161:16912, 1
=====

```

Fig. 7. Sample query and result

As to the evaluation of the approach proposed in this paper, we compare it to PWSD approach in terms of query flexibility. As was already stated earlier, in current approach, the way descriptions are stored in the overlay and the way queries are handled allow for virtually every query underlying database can handle. Given the example query shown in Fig. 5, the way of its handling would be the following: we split the query according to the attributes used in it (in this case it would be 3 groups for attributes **service-type**, **location** and **availability** respectively), find the nodes responsible for each attribute and let them execute their part of the query against their own internal database, which is assumed to be able to handle range queries as well (that is, using operators $>$, $<$ etc.). On the other hand, mechanism proposed in PWSD cannot handle the range part of this query, since, as was described in section 2, each peer stores hash value of the (attribute, value) pair, naturally allowing only exact match queries against data in the original service description.

5 Conclusions

This paper presents the application of P2P technology to solve the problem of efficient services sharing and discovering in a decentralized way. The main advantages of proposed approach are the usage of well-known Chord overlay, which guarantees the correctness and soundness of underlying P2P overlay, and the approach which allows complex, fuzzy and range queries nevertheless keeping the structured overlay approach, while modular framework architecture allows using virtually any P2P overlay and storage mechanism. From the other hand, many of open issues remain, such as introducing distributed transactions for overlay services manipulations, nodes identities that are not tied to the node address (mainly for ad hoc networks) and correct management of services for departed and newly joined nodes.

References

1. Foundation for Intelligent Physical Agents. FIPA Contract Net Interaction Protocol Specification, version H, <http://www.fipa.org/specs/fipa00029/index.html>
2. Foundation for Intelligent Physical Agents. FIPA Recruiting Interaction Protocol Specification, version H, <http://www.fipa.org/specs/fipa00034/index.html>
3. Jennings, N.R.: Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence* 75(2), 195–240 (1995)
4. Pynadath, D.V., Tambe, M., Chauvat, N., Cavedon, L.: Toward Team-Oriented Programming. In: Jennings, N.R. (ed.) *ATAL 1999*. LNCS, vol. 1757, pp. 233–247. Springer, Heidelberg (2000)
5. Kim, M.S., Kim, M.K., Lee, J.T.: Group Situation based Cooperation Model. In: *Proceedings of the 2007 International Conference on Convergence Information Technology (ICCIT 2007)*, pp. 1372–1377 (2007)
6. Schlosser, M.T., Sintek, M., Decker, S., Nejdl, W.: HyperCuP – Hypercubes, Ontologies and Efficient Search on P2P Networks. In: Moro, G., Koubarakis, M. (eds.) *AP2PC 2002*. LNCS (LNAI), vol. 2530, pp. 112–124. Springer, Heidelberg (2003)

7. Ramljak, D., Matijašević, M.: SWSD: A P2P-Based System for Service Discovery from a Mobile Terminal. In: Khosla, R., Howlett, R.J., Jain, L.C. (eds.) KES 2005. LNCS (LNAI), vol. 3683, pp. 655–661. Springer, Heidelberg (2005)
8. Paolucci, M., Sycara, K., Nishimura, T., Srinivasan, N.: Using DAML-S for P2P Discovery. In: International Conference on Web Services, ICWS 2003 (2003)
9. Hu, J., Guo, C., Wang, H., Zou, P.: Web Services Peer-to-Peer Discovery Service for Automated Web Service Composition. In: Lu, X., Zhao, W. (eds.) ICCNMC 2005. LNCS, vol. 3619, pp. 509–518. Springer, Heidelberg (2005)
10. Schmidt, C., Parashar, M.: A Peer-to-Peer Approach to Web Service Discovery. *J. World Wide Web* 7, 211–229 (2003)
11. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup service for Internet applications. In: Proceedings of ACM SIGCOMM 2001, pp. 149–160. ACM (2001)
12. Li, Y., Zou, F., Wu, Z.-D., Ma, F.-Y.: PWS: A Scalable Web Service Discovery Architecture Based on Peer-to-Peer Overlay Network. In: Yu, J.X., Lin, X., Lu, H., Zhang, Y. (eds.) APWeb 2004. LNCS, vol. 3007, pp. 291–300. Springer, Heidelberg (2004)
13. Weiss, G. (ed.): *Muliagent Systems*. The MIT Press, Cambridge (1999)
14. Winoto, W.A., Schwartz, E., Balakrishnan, H., Lilley, J.: The design and implementation of an intentional naming system. In: Proceedings of the Seventeenth ACM Symposium on Operating Systems Principles, vol. 33(5), pp. 186–201. ACM (1999)

Agent-Negotiation of Lot-Sizing Contracts by Simulated Annealing with Part-Way Resets

Mario Ziebuhr, Tobias Buer, and Herbert Kopfer

University of Bremen, Faculty of Business Studies & Economics,
Chair of Logistics, Bremen, Germany,
{ziebuhr,tobias.buer,kopfer}@uni-bremen.de
<http://www.logistik.uni-bremen.de>

Abstract. The distributed multi-level uncapacitated lot-sizing problem is a group decision problem which has to be solved by a set of self-interested and autonomous agents. The agents represent independent companies which have to agree on a joint production plan in a supply chain context. In order to solve the problem we extend a negotiation-based simulated annealing approach introduced by Homberger by a part-way reset procedure. The part-way reset procedure allows a negotiation based search which reaches a deadlock to continue with a different contract proposal and thereby offers a possibility of overcoming disagreements between agents more easily. A benchmark study shows that the approach is competitive on set of 80 medium sized instances from the literature in terms of solution quality, in particular 47 new best-known solutions were computed.

Keywords: negotiation mechanism, lot-sizing, simulated annealing.

1 Collaborative Operations Planning via Agents

Computational supported negotiations between self-interested and autonomous agents are crucial for today's inter-organisational division of labor. We consider agents representing companies in the context of a supply chain. The agents want to negotiate a joint production plan that minimizes the total production costs (global costs), that is, the sum of each agent's production costs. Although the agents have a cooperative attitude, they are still self-interested and want to minimize their own costs (local costs). Furthermore, the agents are not willing to share sensitive information like the degree of capacity utilization or expense ratios (private information). To find a joint production plan, we use collaborative planning based on agent negotiations.

From an economic point of view, the goal of negotiations should be to minimize the global costs [1]. Thereby, the amount of cost savings to share among the agents becomes maximal. Thereafter the savings may be distributed among the agents by means of solution concepts from cooperative game theory. However, we do not focus on the allocation of the savings but on a collaborative

planning enabled by a joint negotiation process with the goal of minimizing the global costs.

Collaborative planning in operations management via agent negotiations is a growing field of interest. The state of the art in collaborative planning is discussed by [2] and [3]. There are many applications of collaborative planning beyond supply chains, e.g., in transport planning, especially enabled via combinatorial auctions [4–8], multi project scheduling [9,10], or machine scheduling [11]. Inter-organisational planning in supply chains is the subject of [12–16], for example. As a representative for supply chain problems, we study a distributed lot-sizing problem.

The studied distributed multi-level uncapacitated lot-sizing problem (DM-LULSP) was introduced by Homberger [17]. It generalizes the well-known multi-level uncapacitated lot-sizing problem (MLULSP) introduced in [18]. The latter is NP-hard for general product structures [19] and therefore computationally challenging. In addition, it features some relevant real world properties of supply chains like a multi-level production structure and a trade-off between inventory holding and setup costs. Coordination of lot-sizing decisions via agent negotiation is difficult because the agents are self-interested with private information. Existing solution approaches for the DMLULSP are based on the meta-heuristics simulated annealing [17], evolutionary strategy [20], and ant colony optimization [21,22].

The goal of this paper is to present an effective extension to the negotiation-based simulated annealing approach introduced by Homberger [17] for the DM-LULSP. The new component is denoted as *part-way reset procedure* (PWR). PWR enables a negotiation based search which reached a deadlock to continue with a different contract proposal and thereby offers a possibility of overcoming disagreements more easily. The benchmark study for our approach shows, that it is able to outperform the most effective approach at present [22] on the set of medium sized instances in terms of solution quality.

The paper is structured as follows. In the following the distributed lot-sizing problem is characterized (Section 2). Section 3 describes the simulated annealing approach and the new part-way reset procedure. Section 4 presents results of a computational experiments and Section 5 concludes the paper.

2 A Distributed Multi-level Uncapacitated Lot-Sizing Problem

2.1 Classical Non-distributed Problem Formulation

Subject to material requirements planning is the determination of the type, the quantity, and the due dates of the required factors of production. This also involves lot-sizing decisions. In a lot-sizing problem the required output per item and per time period is given for a subset of items to produce. Production occurs in batches which are also denoted as lot-sizes. The question is how large these lot-sizes should be, taking into account that larger lot-sizes reduce the total

setup costs of the machines but increase the total stockholding costs whereas smaller lot-sizes have a contrary effect. In addition, there are usually input-output relationships between different items along with other constraints which complicate lot-sizing problems.

The non-distributed or centralized multi-level uncapacitated lot-sizing problem (MLULSP, cf. [18, 23, 24]) can be considered as a special case of the (distributed) DMLUSLP which is why we present it first. In the MLULSP there is a single agent who acts as a central decision maker. The agent is aware of all relevant planning parameters. We are given a set T of possible production periods, $T = \{1, \dots, n\}$, and a set I of items, $I = \{1, \dots, m\}$. The items form a multi-level product structure which is represented in a bill of materials. Final items or final products are assembled of one or more components, components may themselves consist of other components or of raw materials. For each item $i \in I$, we are given a set $\Gamma^+(i) \subset I$ of all direct successors and a set $\Gamma^-(i) \subset I$ of all direct predecessors. Final items are characterized by $\Gamma^+(i) = \emptyset$ and raw materials are characterized by $\Gamma^-(i) = \emptyset$. Furthermore, the production coefficient r_{ij} indicates the required number of item i to produce one unit of item j . Without loss of generality, $r_{ij} = 1$ is assumed. Let s_i, h_i , and t_i be the setup cost, inventory holding cost, and lead time, respectively, per item i ($i \in I$) and per period. For each final item and each period, an exogenous demand d_{it} ($i \in I | \Gamma^+(i) = \emptyset$ and $t \in T$) is given.

In the non-distributed MLULSP, the central decision-making agent has to decide for each item $i \in I$ with $\Gamma^+(i) \neq \emptyset$ and each period $t \in T$ which lot-size x_{it} to produce. Therefore, the endogenous demand d_{it} as well as the inventory l_{it} per item i and per period t have to be determined. Finally, this lot-size decision also includes the setup-decision, i.e., if a production of item i takes place in period t at all ($y_{it} = 1$) or not ($y_{it} = 0$). The mathematical model of the MLULSP is given by the formulas (1) to (8).

$$\min f^{nd}(y) = \sum_{i \in I} \sum_{t \in T} (s_i \cdot y_{it} + h_i \cdot l_{it}) \tag{1}$$

$$\text{s. t.} \quad l_{it} = l_{i,t-1} + x_{it} - d_{it}, \quad \forall i \in I, \forall t \in T, \tag{2}$$

$$l_{i,0} = 0, \quad \forall i \in I, \tag{3}$$

$$l_{it} \geq 0, \quad \forall i \in I, \forall t \in T \setminus \{0\}, \tag{4}$$

$$d_{it} = \sum_{j \in \Gamma^+(i)} r_{ij} \cdot x_{j,t+t_i}, \quad \forall i \in \{j \in I | \Gamma^+(j) \neq \emptyset\}, t \in T, \tag{5}$$

$$x_{it} - M \cdot y_{it} \leq 0, \quad \forall i \in I, \forall t \in T, \tag{6}$$

$$x_{it} \geq 0, \quad \forall i \in I, \forall t \in T, \tag{7}$$

$$y_{it} \in \{0, 1\}, \quad \forall i \in I, \forall t \in T. \tag{8}$$

In the MLULSP, the goal of the central decision-making agent is to minimize his or her (non-distributed) total costs f^{nd} , which are the sum of the setup costs and the stockholding costs for all items $i \in I$ over all periods $t \in T$. The inventory balance (2) ensures that the inventory l_{it} of item i at the end of the

current period t is determined by the inventory of the previous period $t - 1$ and the amount x_{it} produced in the current period minus the demand for item i in the current period. For all items, the inventory of the first period $t = 0$ is zero (3) and for remaining periods non-negative (4). The endogenous demands for the non-final items are determined by (5). These constraints ensure that the production of item j in period $t + t_i$ triggers a corresponding demand d_{it} for all $i \in \Gamma^-(j)$, that is, a demand for each item i preceding item j in the multi-level item structure. Of course, the lot-size x_{it} is non-negative (7). If $x_{it} > 0$, that is, item i is produced in period t , then $y_{it} = 1$, otherwise $y_{it} = 0$. This is enforced by big-M constraints (6).

2.2 Formulation as a Distributed Group Decision Problem

We now drop the assumption that the MLULSP is used for intra-organisational decision-making by a central agent. Instead, the lot-sizing problem is considered as an inter-organisational optimization problem which has to be jointly solved by multiple agents. The extended model is denoted as *distributed multi-level uncapacitated lot-sizing problem* (DMLULSP) and has been introduced by Homberger [17]. In the distributed MLULSP, the set of items I is partitioned and the responsibility to jointly produce the items is assigned to a group of agents A . Each agent might represent an organisational unit of a supply chain, for instance. Agent $a \in A$ is responsible to produce the set of items I_a with $\bigcup_{a \in A} I_a = I$ and $\bigcap_{a \in A} I_a = \emptyset$.

Although the agents have to cooperate to fulfill the overall goals related to the production in the supply chain, each agent is still autonomous and self-interested. Therefore, the individual objective function f_a of agent $a \in A$ is to minimize his or her local costs for producing the items I_a , that is,

$$\min f_a(y) = \sum_{i \in I_a} \sum_{t \in T} (s_i \cdot y_{it} + h_i \cdot l_{it}). \quad (9)$$

Furthermore, the DMLULSP assumes *asymmetric information* regarding the cost parameters s_i and h_i . That is, agent a knows the values of s_i and h_i for all items he or she produces ($i \in I_a$) but not for those items produced by the remaining agents a' ($a' \in A, a' \neq a$) and vice versa. These parameters are considered as private information of agent a because the negotiation power of agent a might be negatively affected if the other agents knew about them. For this reason, agent a does not want to reveal private cost parameters to other agents during collaborative planning. However, we assume the bill of materials is public information due to some kind of common industry knowledge and therefore it is available to all agents (*symmetric information*).

The DMLULSP consists of the constraints (2) to (8) and the objective function (10) which minimizes the total global costs, i.e., the sum of the agent's local costs:

$$\min f(y) = \sum_{a \in A} f_a(y). \quad (10)$$

In the following, we refer to (10) as the global or central cost function of the coalition and the function (9) is denoted local cost function of an individual agent a ($a \in A$). Due to the interdependencies of the items to produce, a change in the global production plan that reduces the total global costs will usually decrease the local costs of some agents while the local costs of other agents are increased. In that sense, the local cost functions of the agents are usually conflicting, which complicates minimizing the global costs. To resolve these possible conflicts and support the agents in agreeing on a joint production plan, we present a collaborative planning approach in the next section.

3 A Simulated Annealing Heuristic with Part-Way Resets

To solve the DMLULSP we propose an approach denoted as simulated annealing with part-way resets (SAR). It is based on the simulated annealing negotiation mechanism introduced by Homberger [17], which we denote as SA. Section 3.1 presents an overview of SAR together with the main components of SA. We extend SA by a *part-way reset procedure* (PWR) introduced in Section 3.2.

3.1 Negotiation of Lot-Sizing Contracts via Simulated Annealing

Simulated annealing is a metaheuristic based on local search [25] which is frequently used to solve discrete optimization problems. In order to escape from local optima simulated annealing allows also moves that increase the objective function value (in case we consider a minimization problem). Whether an increasing move is performed depends on a probability measure which in turn depends on the objective function value of the solution and in particular a *temperature* parameter. At first, the temperature is set to a high value which decreases (*anneals*) as the search advances. Higher temperature values cause a higher probability of accepting increasing moves.

An overview of SAR is given by Alg. 1. Due to the extension of PWR described in Section 3.2 two phases *store* and *reset* were added. We now describe the most important components of SA and refer the reader for details to [17].

Encoding and Decoding of a Solution as a Contract. Although a complete solution for the (D)MLULSP covers the setup-decisions, the lot-sizes, and the inventory holding for each item and each period, it is possible to infer from the setup-decisions the remaining decisions. That is, from a given y_{it} ($i \in I, t \in T$) the values of the variables x_{it} and l_{it} can be derived [24]. Therefore, we simply denote the setup decisions y_{it} for the DMLULSP as solution.

In SA a solution is encoded and represented by a *contract*. A contract c is a $|I| \times |T|$ binary matrix, where $c_{it} = 1$ represents a *possible but not mandatory* production of item i in period t , i.e., $c_{it} = 1 \Rightarrow y_{it} = 1 \wedge y_{it} = 0$ and $c_{it} = 0 \Leftrightarrow y_{it} = 0$. The decoding procedure for an encoded solution is described in [26].

Algorithm 1. The negotiation mechanism SAR (cf. [1, 13, 17])

Data: problem data, $r_{max}, \gamma, k, \Delta, T_a^0(a \in A), T^{end}$

```

1 mediator   phase  $\leftarrow store$ 
2 mediator   negotiation round counter  $r \leftarrow 0$ 
3 mediator   generate initial contract  $c$  randomly
4 each  $a \in A$    evaluate contract  $c$  with local objective function  $f_a(c)$ 
5 each  $a \in A$    compute cooling schedule  $\tau_a \leftarrow \sqrt{(r_{max}-1)T^{end}/T_a^0}$ 
6 while  $r \leq r_{max}$  do
7   mediator    $r \leftarrow r + 1$ 
8   mediator   generate a new proposal  $c' \leftarrow N(c)$ 
9   each  $a \in A$    evaluate proposal  $c'$  with local objective function  $f_a(c')$ 
10  each  $a \in A$    accept  $c'$  with probability  $P_a(c, c', r)$ 
11  if all agents accept proposal  $c'$  then
12    mediator   update the mutually accepted contract  $c \leftarrow c'$ 
13    each  $a \in A$    update temperature  $T_a^r \leftarrow \tau_a \cdot T_a^{r-1}$ 
14  end
15  if  $r \bmod (\rho \cdot r_{max}) = 0$  then
16    if  $|C| = \gamma \wedge \mathbf{phase} = store \wedge deadlock$  then
17      mediator   phase  $\leftarrow reset$ 
18    end
19    mediator    $c \leftarrow PartwayReset(c', T, \Delta, k, \mathbf{phase})$ 
20  end
21 end
22 return  $c$  mutually accepted contract

```

Initial Contract. The first contract c which is used as a starting point for further negotiations is generated randomly by the mediator (cf. Alg. 1, line 3). For the initial contract, with a probability of 50 percent c_{it} is set to 1 and to 0 otherwise ($i \in I, t \in T$). There is one exception, namely in the first period with a demand a production is always possible for all items. Because there are no capacities in the lot-sizing problem at hand, this ensures that a contract can always be decoded to a feasible solution.

Generate New Contract Proposal. In line 8 of Alg. 1 the mediator generates a new contract proposal c' by flipping a randomly chosen element from c . That is, the neighborhood $N(c)$ of the currently accepted contract c is defined as the set of contracts that can be obtained by flipping a single element of c , i.e., if $c_{it} = 1$ then $c'_{it} = 0$ and if $c_{it} = 0$ then $c'_{it} = 1$.

Stochastic Contract Acceptance Criterion. In line 10 of Alg. 1 agent $a \in A$ always prefers a contract proposal c' to the up to now accepted contract c if the agent's local cost decrease ($f_a(c') \leq f_a(c)$). Nevertheless, if c' leads to higher local costs ($f_a(c') > f_a(c)$) agent a prefers c' to c with probability $P_a(c, c', r)$

where r is the current negotiation round:

$$P_a(c, c', r) = \begin{cases} 1 & \text{if } f_a(c') \leq f_a(c), \\ \exp\left(\frac{f_a(c) - f_a(c')}{T_a^r}\right) & \text{otherwise.} \end{cases} \quad (11)$$

Formula (11) represents the metropolis acceptance criterion for moves from simulated annealing. Here, each agent $a \in A$ has an individual temperature T_a^r .

Agent's Cooling Schedule. In each negotiation round r the temperature T_a^r of agent a decreases. A decreasing temperature implies that the probability to accept a non-improving contract declines as well. In line 13 of Alg. 1 the geometric cooling schedule of (12) is used. In that schedule, τ_a is a constant factor depending on the end temperature T^{end} to which the agent's individual temperatures T_a^r should anneal.

$$T_a^r = \tau_a \cdot T_a^{r-1} \text{ with } \tau_a = \sqrt[r_{max}-1]{\frac{T^{end}}{T_a^0}} \quad (12)$$

T^{end} and T_a^0 are given parameters of the search. To compute appropriate T_a^0 values of each agent $a \in A$ Fink [13] suggested trial runs. We acted on that suggestion taking into account the modifications of [17]. In a trial run, roughly speaking, a negotiation of agent a with the mediator is simulated and the success and failure of contract proposals are measured and the initial temperature values T_a^0 are computed.

3.2 A Part-Way Reset Procedure

We now extend the SA approach by a *part-way reset procedure* (PWR). Therefore, we maintain some contracts already found during search together with the involved temperature parameters of each agent. If the search does not progress in a satisfactory way, the search may reset to an earlier solution and continue from there with some modifications of the agents' temperature values. Unlike restart metaheuristics like GRASP (greedy randomized restart procedure, e.g. [27]) a new solution is not constructed from scratch but the search is only set back in part. Therefore we denote this component as part-way reset procedure.

The idea for a part-way reset is motivated by an analysis of the computational results of [17]. There appear to be two shortcomings:

1. The best contract is already found after about fifty to seventy percent of the maximal negotiation rounds r^{max} .
2. All agents have to mutually agree on a contract, therefore only a single minor agent might veto a globally improving solution.

By the part-way reset procedure we try to overcome both points. A reset to an earlier contract allows the search to explore different areas in the contract space

in the remaining negotiation rounds and therefore use these more effectively. In addition to a reset of the contract, we raise the temperatures of some randomly selected agents A' ($A' \subset A$). This implies for the agents in A' that the probability to accept a non-improving contract increases. From the point of view of an agent in A' this is a clear handicap because his or her power of veto weakens. Therefore, we make sure that a raise of the temperature occurs for all agents ($a \in A$) only once during the whole search process.

The PWR relies on a candidate set C of γ potential reset points. A reset point R_j ($1 \leq j \leq \gamma$) consists of an encoded solution c' found in negotiation round r together with the values of the temperature parameters used by each agent to reach this solution, i.e., $R_j := (r, c', T_1^r, \dots, T_{|A|}^r)$. Note, the encoded solution c' may not be accepted by all agents. In short, the candidate set of reset points is defined as follows:

$$C = \{R_j\} \text{ with } 1 \leq j \leq \gamma \quad \text{and} \quad R_j = (r, c', T_1^r, \dots, T_{|A|}^r) \quad (13)$$

PWR consists of two phases. In the *first phase* of PWR (cf. Alg. 2, lines 2–4) potential reset points are added to the candidate set C . According to Alg. 1, every $\rho \cdot r_{max}$ negotiation rounds a reset point R is added to C . Both parameters, the maximum number of negotiation rounds r_{max} and the fraction ρ are external parameters. Here, we generate a reset point every $\rho \cdot r_{max} = 20,000$ iterations.

Algorithm 2. Part-way reset procedure

Data: $C, c', T, \Delta, k, \text{phase}$

```

1 if phase = store then
2   mediator   generate a reset point  $R \leftarrow (r, c', T_1, \dots, T)$ 
3   mediator    $C \leftarrow C \cup \{R\}$ 
4   mediator   if  $|C| = \gamma$  then phase  $\leftarrow$  reset
5 else if phase = reset then
6   mediator   randomly select a reset point  $R \in C$  and update  $C \leftarrow C \setminus \{R\}$ 
7   mediator   replace the current contract  $c \leftarrow c(R)$ 
8   mediator   randomly select a subset  $A' \subset A$  with cardinality  $k$ 
9   each agent  $a \in A$    update temperature  $T_a \leftarrow T_a(R)$ 
10  each agent  $a \in A'$    increase temperature  $T_a \leftarrow T_a \cdot \Delta$ 
11  return  $c$ 
12 end

```

In the *second phase* of PWR (cf. Alg. 2, lines 6–11), the usage of a reset point is allowed. According to Alg. 1, line 16, the second phase may only start after γ reset points have been generated and the negotiation reaches a *deadlock*. A *deadlock* is reached, if it is not possible to find a new mutually accepted contract after a defined number of negotiation rounds. In the second phase, a reset point R is randomly selected from C and the current contract c is replaced by the contract of the reset point $c(R)$ and the temperature parameter T_a^r of

each agent $a \in A$ in the current round r is reset to $T_a(R)$. Furthermore, the values for the temperature of k agents in A are raised by an additional factor Δ . Therefore, the probability of these k agents to accept an inferior contract increases and the chance to escape a local optimum originated by a minority of agents increases as well. After the second phase started, the search cannot return to the first phase. That is, in phase one the candidate set C is filled until γ reset points are achieved and afterwards emptied in phase two as long as the maximum number of negotiation rounds r_{max} are not completed.

Note, with our test instances from the literature, a mutual accepted contract is always found before a reset of the search is performed. In phase one, SAR tries to improve a jointly accepted contract. In phase two resets of the search are allowed and SAR may also try to improve a contract which is temporarily *not* jointly accepted.

4 Computational Results

We use a computational benchmark study to evaluate SAR. Section 4.1 describes the setup of the benchmark study. In Section 4.2 we analyse the effect of PWR on a subset of instances and in Section 4.3 we compare the performance of SAR to results from the literature.

4.1 Set-up of Computational Study

120 benchmark instances are used in this study¹. Our study is based on the 40 medium-sized instances for the MLULSP [24,28] where a single agent ($|A| = 1$) is responsible to produce all items. We use these MLULSP instances as a reference scenario (instance group m1). Based on these MLULSP instances, [17] introduced instances for the DMLULSP. From these, we use the 40 medium-sized instances for the DMLULSP with two agents (instance group m2), and 40 instances with five agents (instance group m5).

As essential evaluation criteria, we consider the gap of the distributed solution y computed by SAR for a given instance, to the best-known non-distributed solution y^{bk} from the literature as suggested by [14,15]. The percentage gap $\mathcal{G}(y)$ is calculated as:

$$\mathcal{G}(y) = \frac{f(y) - f^{nd}(y^{bk})}{f^{nd}(y^{bk})} 100. \quad (14)$$

The best-known values $f^{nd}(y^{bk})$ have been gathered from [17,24,26,29,30]. The total cost of an optimal MLULSP solution is a lower bound for the total cost of an optimal solution of the DMLULSP. However, it is unknown, if the reported solutions for the medium sized instances are optimal. Therefore, we should expect $\mathcal{G}(y) \geq 100\%$, which is however not guaranteed.

SAR was implemented in JAVA (JDK 1.7) and the computational experiments were executed on a Windows 7 personal computer with Intel Core i7-2600 processor (3.4 GHz and 16 GB of main memory).

¹ Instances are available at <http://www.dmlulsp.com>

4.2 Effect of Part-Way Reset Procedure

For the first test, we compare the reimplemented SA to SAR (i.e., SA extended by PWR) in order to evaluate the effectiveness of the reset procedure PWR. The results of this comparison are shown in Table 1. The table shows results for three instance groups m1, m2, and m5. For each instance group, only a subset of ten instances is computed. The table lists the applied parameter values for the number of negotiation rounds r_{max} , the initial acceptance probability for a non-improving solution P_{init} , the temperature increment Δ , and the storage frequency ρ of reset points. As can be seen from the two rightmost columns,

Table 1. Comparison of SAR with and without part-way reset procedure

group	instances	r_{max}	P_{init} (%)	Δ	ρ (%)	mean \mathcal{G} (%)	
						SA	SAR
m1	15-24	$4 \cdot 10^5$	60	10	0.25	0.845	0.700
m2	15-24	$4 \cdot 10^5$	45	2.5	0.25	2.421	1.774
m5	15-24	$4 \cdot 10^5$	90	5	0.25	21.862	15.392

under the given test conditions, the PWR extension is able reduce the gap to the central best-known solution for all three instance groups. The effect reinforces, at least for these instances, if the distributed nature of the problem increases. The achieved reduction for group m1, m2, and m3 is significant with about 21 percent, 27 percent, and 30 percent, respectively.

4.3 Comparison with Heuristics from the Literature

We compare the performance of SAR to five state of the art heuristics from the literature designed for the DMLULSP. The five approaches are denoted as ES, ACO, CACM, HACM, and SA. ES is based on an evolutionary strategy and is presented in [20], ACO is discussed by [21]. ACO is based on the ant colony metaheuristic. Similarly, CACM and HACM [22] are both based on the ant metaheuristic, however, both approaches use a more sophisticated encoding strategy on which the ant search graph is based. In contrast to the other approaches, HACM (hierarchical ant colony optimization) is not applicable to general DMLULSP instances but to instances with a specific multi-level item structure that enables hierarchical planning. SA is based on simulated annealing [17], our approach SAR is an extension of the SA approach. The SA results are extracted from the literature as well and not recomputed.

Table 2 shows the gap \mathcal{G} averaged over the 40 instances of group m2 and m5, respectively. For instance group m2, the SAR outperforms all other approaches. However, on the instance group m5 with five agents, SAR cannot outperform the ant approaches. However, the results show again, the PWR extension increases the average solution quality of SA, because SAR outperforms SA on both groups, m2 and m5.

Table 2. Average gap \mathcal{G} for SAR and five heuristics from the literature (cf. [22])

group	$ A $	\mathcal{G}					
		ES	SA	ACO	CACM	HACM	SAR
m2	2	47.23	7.11	5.73	1.98	3.29	1.13
m5	5	96.15	12.13	8.15	8.06	8.06	10.97

Table 3. Computational results for CACM, HACM (cf. [22]) and SAR

no.	central	m2, $ A = 2$ agents			m5, $ A = 5$ agents		
	best-known	CACM	HACM	SAR	CACM	HACM	SAR
1	194571	198654.45	201111.70	198057.95	207655.95	208247.35	205296.95
2	179762	180144.25	187453.30	181203.85	198732.35	201047.75	217802.15
3	165110	167159.30	171486.90	167226.90	174637.30	174803.05	178324.55
4	155938	156130.95	157343.20	156184.20	168125.95	169405.35	186271.70
5	201226	201436.75	205828.80	207900.75	215672.95	217148.80	215828.85
6	183219	183316.80	183316.80	184874.00	203307.80	203307.80	206855.15
7	187790	189432.95	192141.90	189181.85	204519.15	206384.90	193221.55
8	136462	140267.00	143273.10	139788.15	154127.50	155005.80	144373.50
9	161304	168115.90	171232.00	163590.55	168236.50	171153.90	168651.05
10	186597	188307.30	188832.50	187654.10	202756.50	204165.30	198237.60
11	342916	347533.80	357774.10	346877.90	371034.90	371768.95	356741.00
12	340686	350363.75	356713.70	347745.95	374680.00	377264.80	361902.80
13	292908	296308.90	297279.35	294518.80	312728.90	312516.70	310901.15
14	378845	383953.00	393565.40	391933.55	418806.15	421202.30	436125.55
15	354919	363276.35	365881.15	360608.60	374513.25	374964.25	378290.50
16	346313	355522.90	362010.00	350535.25	391312.00	392701.40	402815.15
17	325212	333256.15	340158.25	334694.05	360186.00	362002.10	340470.85
18	411997	421956.45	422673.80	415172.85	467870.25	470057.20	449555.65
19	385939	397906.50	403863.50	390126.90	427922.75	429155.40	411996.60
20	390194	398145.05	406909.50	399311.90	447712.55	450545.85	445091.95
21	148004	148697.75	153633.55	148126.10	153985.00	154732.70	177796.00
22	185161	192901.15	195513.15	185469.25	198846.05	200145.65	200172.70
23	197695	201034.60	202926.20	199737.70	204575.00	204627.00	233723.85
24	185542	188634.85	190227.60	188060.20	197505.30	198291.65	192770.20
25	160693	160924.90	161290.90	160692.90	170716.95	172166.70	168013.80
26	192157	197132.00	197589.55	194746.90	203624.90	205281.35	197233.05
27	184358	186187.90	187520.35	184578.25	190854.45	191636.40	195043.70
28	136757	138717.25	139620.70	137555.90	142564.05	143993.00	151239.40
29	161457	161967.00	161967.00	161549.00	169077.40	169112.70	177858.00
30	166041	169387.00	172662.30	167803.60	181454.00	183848.25	175226.70
31	344970	353513.20	357377.20	347226.10	373222.00	376412.35	415677.45
32	289846	294543.90	295794.25	290675.05	309364.45	309271.30	310046.50
33	352634	369629.50	369877.00	352902.55	384679.85	388249.85	391213.50
34	337913	344460.85	345452.15	344659.65	362379.50	363750.50	385450.10
35	356323	363559.75	364774.25	359197.10	385051.35	385597.30	487519.55
36	319905	327621.70	331926.50	323251.30	342949.55	345287.60	349326.85
37	411338	433067.10	431887.60	413189.95	459952.45	460670.15	446384.10
38	366848	377363.70	379710.05	371951.85	396312.90	397393.40	390989.10
39	401732	416743.90	418036.95	406274.85	430166.95	430335.75	578394.75
40	305011	306785.75	308950.90	307842.50	318313.65	319545.80	332210.90
mean		268851.56	271889.68	266316.97	285503.36	286829.96	294126.11
median		247990.33	250811.53	249287.90	262518.70	263210.05	271885.18

The objective function values computed by SAR for each of the 80 instances are given in Table 3. The second column states the best-known solution for the non-distributed MLULSP (group m1) according to the literature. Columns three to five state the results for the m2 instances and columns six to eight state results for the m5 instances. The results were computed by the heuristics CACM, HACM (see [22]) and the SAR, respectively. The objective function values of the best-known solutions are highlighted in bold.

From Table 3 it can be seen, that SAR improves 30 out of 40 best-known solutions for instances group m2 and 17 out of 40 best-known solutions for instance group m5. In a direct comparison with the heuristics CACM and HACM, SAR outperforms CACM on 30 out of 40 m2-instances and on 17 out of 40 m5-instances; SAR outperforms HACM on 38 out of 40 m2-instances and on 19 out of 40 m5-instances. All in all, SAR seems competitive with respect to the obtained solution quality to the state of the art heuristics from the literature. While SAR seems superior on two-agent instances, CACM appears to be superior for five-agent instances. We believe this is caused by the random selection of the subset of agents whose temperatures are increased (cf. Alg. 2, line 8), because the chance to increase the temperature of an agent that is actually causing a deadlock decreases with a higher number of involved agents.

With respect to a runtime comparison of the different approaches it has to be mentioned that SAR is clearly slower than SA, CACM, and HACM. On average, SA needs 42 seconds [7] per DMLULSP instance while SAR requires 296 seconds. For the most part, we attribute the longer computing times to a less efficient implementation of SAR. In any case, the PWR mechanism itself is undemanding with respect to implementation requirements. However, it should be noted that the compared approaches generate for each instance the same number of solutions, i.e. the opportunity to identify a good solution is equal for the tested heuristics.

5 Conclusions and Outlook

This paper studied an inter-organisational lot-sizing problem which has to be solved by a set of self-interested and autonomous agents. The problem is denoted as distributed multi-level uncapacitated lot-sizing problem (DMLULSP). To solve it, a negotiation approach based simulated annealing introduced by Homberger [17] was extended by a *part-way reset procedure* (PWR). The idea is to overcome disagreements between agents more easily by resetting the search once in a while to earlier solutions and discriminating some agents randomly during the negotiation process. Compared to the other agents, a discriminated agent has higher probability of accepting a contract which does not decrease his or her local costs. SAR outperforms the best-known heuristics from the literature on 30 out of 40 medium sized instances with two agents and on 17 out of 40 medium sized instances with five agents. Future research should focus on developing an intelligent schema to adaptively parameterize PWR – in particular the selection of agents whose temperature values are increased – so that it computes competitive results for larger lot-sizing instances.

References

1. Klein, M., Faratin, P., Sayama, H., Bar-Yam, Y.: Negotiating complex contracts. *Group Decision and Negotiation* 12(2), 111–125 (2003)
2. Frayret, J.M.: A multidisciplinary review of collaborative supply chain planning. In: *IEEE Int. Conf. on Systems, Man and Cybernetics*, 2009, pp. 4414–4421 (2009)
3. Stadler, H.: A framework for collaborative planning and state-of-the-art. *OR Spectrum* 31(1), 5–30 (2009)
4. Pankratz, G.: Analyse kombinatorischer Auktionen für ein Multi-Agentensystem zur Lösung des Groupage-Problems kooperierender Expeditionen. In: Inderfurth, K., Schwödiauer, G., Domschke, W., Juhnke, F., Kleinschmidt, P., Wäscher, G. (eds.) *Operations Research Proceedings 1999*, pp. 443–448. Springer, Berlin (2000)
5. Berger, S., Bierwirth, C.: Solutions to the request reassignment problem in collaborative carrier networks. *Transportation Research Part E: Logistics and Transportation Review* 46(5), 627–638 (2010)
6. Buer, T., Kopfer, H.: Shipper decision support for the acceptance of bids during the procurement of transport services. In: Böse, J.W., Hu, H., Jahn, C., Shi, X., Stahlbock, R., Voß, S. (eds.) *ICCL 2011*. LNCS, vol. 6971, pp. 18–28. Springer, Heidelberg (2011)
7. Buer, T., Kopfer, H.: A Pareto-metaheuristic for a bi-objective winner determination problem in a combinatorial reverse auction. *Computers & Operations Research* (in press, 2013), doi: 10.1016/j.cor.2013.04.004
8. Lang, F., Fink, A.: Negotiating in dynamic environments: time-efficient automated negotiations by means of combinatorial auctions. *Evolving Systems* 3(3), 189–201 (2012)
9. Homberger, J.: A (μ, λ) -coordination mechanism for agent-based multi-project scheduling. *OR Spectrum* 34(1), 107–132 (2012)
10. Fink, A., Homberger, J.: An ant-based coordination mechanism for resource-constrained project scheduling with multiple agents and cash flow objectives. *Flexible Services and Manufacturing Journal* 25(1-2), 94–121 (2013)
11. Lang, F., Fink, A.: Collaborative single and parallel machine scheduling by autonomous agents. In: *Int. Conf. on Collaboration Technologies and Systems (CTS)*, pp. 76–83 (2012)
12. Ertogral, K., Wu, S.: Auction-theoretic coordination of production planning in the supply chain. *IIE Transactions* 32(10), 931–940 (2000)
13. Fink, A.: Supply chain coordination by means of automated negotiation between autonomous agents. In: *Multiagent-based supply chain management*, pp. 351–372. Springer, Berlin (2006)
14. Dudek, G., Stadler, H.: Negotiation-based collaborative planning between supply chains partners. *European Journal of Operational Research* 163(3), 668–687 (2005)
15. Dudek, G., Stadler, H.: Negotiation-based collaborative planning in divergent two-tier supply chains. *International Journal of Production Research* 45(2), 465–484 (2007)
16. Drechsel, J., Kimms, A.: Cooperative lot sizing with transshipments and scarce capacities: solutions and fair cost allocations. *International Journal of Production Research* 49(9), 2643–2668 (2011)
17. Homberger, J.: Decentralized multi-level uncapacitated lot-sizing by automated negotiation. *4OR: A Quarterly Journal of Operations Research* 8, 155–180 (2010)
18. Yelle, L.: Materials requirements lot sizing: A multilevel approach. *International Journal of Production Research* 17(3), 223–232 (1979)

19. Arkin, E., Joneja, D., Roundy, R.: Computational complexity of uncapacitated multi-echelon production planning problems. *Operations Research Letters* 8(2), 61–66 (1989)
20. Homberger, J.: A generic coordination mechanism for lot-sizing in supply chains. *Electronic Commerce Research* 11(2), 123–149 (2011)
21. Homberger, J., Gehring, H.: A pheromone-based negotiation mechanism for lot-sizing in supply chains. In: *Proceedings of the 44nd Hawaii International Conference on System Sciences*, pp. 1–10. IEEE Computer Society, Washington, DC (2010)
22. Buer, T., Homberger, J., Gehring, H.: A collaborative ant colony metaheuristic for distributed multi-level uncapacitated lot-sizing. *International Journal of Production Research* (in press, 2013), doi: 10.1080/00207543.2013.802822
23. Steinberg, E., Napier, H.: Optimal multi-level lot sizing for requirements planning systems. *Management Science* 26(12), 1258–1271 (1980)
24. Dellaert, N., Jeunet, J.: Solving large unconstrained multilevel lot-sizing problems using a hybrid genetic algorithm. *International Journal of Production Research* 38(5), 1083–1099 (2000)
25. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. *Science* 220, 671–680 (1983)
26. Homberger, J.: A parallel genetic algorithm for the multilevel unconstrained lot-sizing problem. *INFORMS Journal on Computing* 20(1), 124–132 (2008)
27. Resende, M.G., Ribeiro, C.C.: Greedy randomized adaptive search procedures: Advances and applications. In: Gendreau, M., Potvin, J.Y. (eds.) *Handbook of Metaheuristics*, 2nd edn., pp. 281–317. Springer, Berlin (2010)
28. Afentakis, P., Gavish, B.: Optimal lot-sizing algorithms for complex product structures. *Operations Research* 34(2), 237–249 (1986)
29. Pitakaso, R., Almeder, C., Doerner, K., Hartl, R.: A max-min ant system for unconstrained multi-level lot-sizing problems. *Computers & Operations Research* 34(9), 2533–2552 (2007)
30. Xiao, Y., Kaku, I., Zhao, Q., Zhang, R.: A reduced variable neighborhood search algorithm for uncapacitated multilevel lot-sizing problems. *European Journal of Operational Research* 214(2), 223–231 (2011)

AbstractSwarm – A Generic Graphical Modeling Language for Multi-Agent Systems

Daan Apeldoorn

Dept. of Computer Science, FernUniversität in Hagen,
58084 Hagen, Germany
Daan.Apeldoorn@FernUni-Hagen.de

Abstract. Many different approaches exist for modeling multi-agent systems: The variety ranges from frameworks dedicated to a specific problem domain to very flexible systems similar to text-based or graphical programming languages. While problem domain specific frameworks have an obvious lack of flexibility (and thus can only be configured for a small set of similar problems), the more flexible systems enforce detailed modeling, in which many details (e.g. the agent behavior) have to be implemented manually. As a consequence, the results obtained from such systems are often very heterogeneous and incomparable across problem domains. In this paper, a graphical modeling language is introduced that focuses on a common base for multi-agent systems. By strictly separating the modeling of agent tasks from the agent behavior, the same agent models can be reused for different scenarios and the obtained results will be comparable across problem domains.

Keywords: graphical modeling language, logistics, multi-agent systems.

1 Introduction

The modeling and simulation of multi-agent systems (MASs) is supported by various approaches: Some systems are dedicated to a specific problem domain and provide configuration possibilities for adaption to a concrete problem (e.g. [1]). Other systems feature text-based or graphical programming of the environment and agent behavior through a framework (e.g. [7]) or an appropriate programming language (e.g. [4], [6]). These systems offer high flexibility, but the implementation of a MAS must be achieved by both defining the problem domain and the agent behavior (e.g. how agents have to act in the context of the specified problem domain). Furthermore, the obtained results can be of very different kinds, depending on how the MAS was modeled. Thus, it can be hard to find a general evaluation criteria across problem domains in order to compare the quality of differently modeled MASs (e.g. for the evaluation of the same agent models in different problem domains).

With “AbstractSwarm”, a graphical modeling language is introduced that allows the intuitive decomposition of complex problems and their modeling as MASs for a wide range of problem domains. The description of the agent tasks

(what to do) is strictly separated from the definition of agent behavior (how to do it). Thus, generic agent models can be used for a variety of different problems. The modeling approach suggests the simulation results to be represented as Gantt charts [9] and thereby allows evaluation and comparison of agent behavior across different problem domains. A software tool which supports the modeling with AbstractSwarm by only allowing the creation of syntactically correct models has been developed.

The paper is organized as follows: In Section 2, the graphical modeling language is introduced in detail. Section 3 then concludes with an outlook of how results can be obtained as Gantt charts from a discrete simulation of MASs modeled with AbstractSwarm.

2 Syntax and Semantics

In AbstractSwarm, MASs are modeled as so-called “AbstractSwarm graphs”. These graphs allow to describe MASs in an abstract way, as their semantics represent a generalization of common properties of MASs. As a result, the same agent models can be used for any problem described by an AbstractSwarm graph, regardless of the problem domain.

To generalize problems which are intended to be modeled and solved by MASs, concepts from the field of logistics serve as a common base. For this reason, a variety of prototypical logistical problems (e.g. transportation/transshipment, vehicle routing and flow maximization problems; see [3], [5] and [8]) and their underlying questions have been examined. Syntax and semantics of AbstractSwarm graphs reflect this approach to MASs.¹

In addition, a special syntax element is offered to decompose complex problems into so-called “perspectives”. This directs the modeling process from problem to MAS in an intuitive way.

An AbstractSwarm graph is defined as a tuple

$$G := (\Psi, \mathcal{E}) \quad (1)$$

where Ψ is a set of perspectives and \mathcal{E} is a set of edges. Every perspective $\psi \in \Psi$ contains a set $S_{TC} := S_{TA} \cup S_{TS}$ of so-called “component types” where S_{TA} is a set of agent types and S_{TS} is a set of station types (see section 2.1). The set $\mathcal{E} := E_V \cup E_T \cup E_P$ represents relations among the component types where E_V is a set of «visit» edges, E_T is a set of «time» edges and E_P is a set of «place» edges (see section 2.2).

In sections 2.1, 2.2 and 2.3 the basic component types, their relations and the attributes that can be assigned to component types are introduced. Section 2.4 explains the decomposition of complex problems into perspectives.

¹ A description of the problem classes that influenced the language design would go beyond the scope of this paper and can be found in [2] or in the online appendix: http://www.mosd.net/apeldoorn/AbstractSwarm_Appendix.pdf

2.1 Agents and Stations

From an abstract point of view, there are two basic elements that are important to describe when modeling MASs: Agents and the environment within which they act. While agents can be regarded as the moving or “mobile” components of a system, the environment has a more stationary or “immobile” character. This leads to the two basic component types of the graphical modeling language: “agent types” and “station types”.

Agents and stations are modeled with the agent type and the station type syntax elements. These elements represent a defined number of components of the respective type. Agent types and station types are the nodes of the graphical model.

Agent Types. An agent type $T_A \in S_{T_A}$ is defined as a tuple

$$T_A := (t_A, P_A, A) \quad (2)$$

where t_A is a type name, P_A is a set of attributes (defining the properties of the agents) and $A \neq \emptyset$ is a set of agents with every agent $a \in A$ of type T_A .

Agent types are graphically represented by circles with a centered label that corresponds to the type name t_A . A cardinality $c_A := |A|$ at the top left corner indicates how many agents of an agent type T_A exist. Since the role of single agents typically differs from the role of agents occurring in larger quantities, a syntactical distinction is made by one single circle in case of one agent, or by two or three overlapping circles in case of two or more agents.

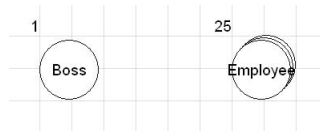


Fig. 1. Example of agent types: one agent of type “Boss” and 25 agents of type “Employee” (Source: AbstractSwarm software tool, adapted from [2], p. 11)

Station Types. Analogously to an agent type, a station type $T_S \in S_{T_S}$ is defined as a tuple

$$T_S := (t_S, P_S, S) \quad (3)$$

where t_S is a type name, P_S is a set of attributes (defining the properties of the stations) and $S \neq \emptyset$ is a set of stations with every station $s \in S$ of type T_S .

Station types are graphically modeled with squares: One or more overlapping squares represent one or more stations. A centered label shows the type name t_S and a cardinality $c_S := |S|$ at the top left corner indicates how many stations of type T_S exist.

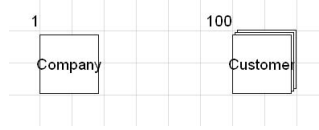


Fig. 2. Example of station types: one station of type “Company” and 100 stations of type “Customer” (Source: AbstractSwarm software tool, adapted from [2], p. 12)

2.2 Relations

Relations among component types are defined by edges. There are three different kinds of edges that model visiting, spatial or temporal dependencies.

«visit» Edges. A «visit» edge $e_V \in E_V$ is an undirected, unweighted edge that expresses a visiting dependency between an agent type $T_A \in S_{T_A}$ and a station type $T_S \in S_{T_S}$. It is defined as a tuple

$$e_V := (\{T_A, T_S\}, b) \tag{4}$$

where $b \in \{\text{True}, \text{False}\}$ is a Boolean value that determines whether the edge is bold.

A «visit» edge is graphically represented by a single line between an agent type and a station type. It states that the agents $a \in A$ of agent type T_A have to visit the stations $s \in S$ of the connected station type T_S . If an agent a enters a station s , a “visit event” is triggered on both components a and s .

If the «visit» edge is bold, it additionally determines that every $a \in A$ starts at one of the connected stations $s \in S$. If an agent type T_A has more than one bold «visit» edge, every $a \in A$ of type T_A starts at an arbitrary station s of the station types that are connected with a bold «visit» edge.

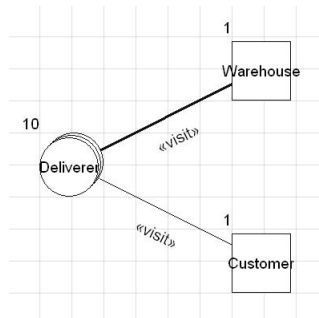


Fig. 3. Fig. 3. Example of «visit» edges: The stations of type “Warehouse” and type “Customer” can be visited by agents of type “Deliverer”. Deliverers start at the warehouse. (Source: AbstractSwarm software tool, adapted from [2], p. 13)

«place» Edges. The «place» edges express spatial dependencies. A «place» edge $e_P \in E_P$ is a directed or an undirected, weighted edge between two station types $T_{S_1}, T_{S_2} \in S_{T_S}$. It is defined as a tuple

$$e_P := (\{T_{S_1}, T_{S_2}\}, D_P, w_P) \quad (5)$$

where $D_P \in \{\{T_{S_1}\}, \{T_{S_2}\}, \emptyset\}$ is a set containing the station type to which the edge is directed (with $D_P = \emptyset$ in case e_P is undirected) and $w_P \in \mathbb{N} \cup \{0\}$ is the weight.

A «place» edge is graphically represented by a single line between two station types with the weight w_P shown inside a rhombus centered on the line. It states that there is a route between every station $r \in S_1$ of station type T_{S_1} and every station $s \in S_2$ of the connected station type T_{S_2} . The distance between the stations is defined by w_P .

If a «place» edge is directed, it additionally states that stations of the connected types must be visited in a specific order, following the direction of the edge. For example, if $D_P = \{T_{S_2}\}$ then a station $r \in S_1$ of station type T_{S_1} must be visited by an agent a , before a station $s \in S_2$ of station type T_{S_2} can be visited by a .

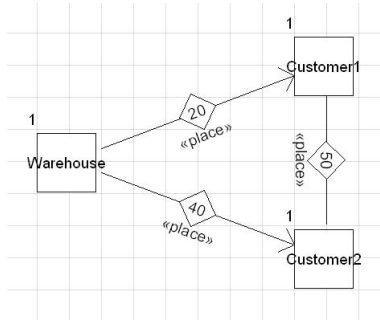


Fig. 4. Example of «place» edges: A company supplies two customers at a distance of 20 and 40 units. The distance between the customers is 50 units. A visit of both customers must be preceded by a visit of the warehouse. (Source: AbstractSwarm software tool, adapted from [2], p. 14).

«time» Edges. The «time» edges express temporal dependencies. A «time» edge $e_T \in E_T$ is a directed or undirected, weighted edge between any two component types $T_{C_1}, T_{C_2} \in S_{T_C}$. It is defined as a tuple

$$e_T := (\{T_{C_1}, T_{C_2}\}, D_T, L, w_T) \quad (6)$$

where $D_T \in \{\{T_{C_1}\}, \{T_{C_2}\}, \emptyset\}$ is a set containing the component type to which the edge is directed (with $D_T = \emptyset$ in case e_T is undirected). The set $L \in \{\{T_{C_1}\}, \{T_{C_2}\}, \{T_{C_1}, T_{C_2}\}, \emptyset\}$ contains the component types to which e_T is connected with a logical “AND relation”. If $L \neq \emptyset$ then either e_T must be undirected

or $D_T = L$. If $L = \emptyset$ then e_T does not have any AND relations. The weight of the edge is defined as $w_T \in \mathbb{N} \cup \{0\}$.

A «time» edge is graphically represented by a double line between two agent types, two station types, or an agent type and a station type. The weight is shown inside a rhombus centered on the double line. A «time» edge e_T expresses a temporal relation between the component types T_{C_1} and T_{C_2} : The weight w_T specifies the number of time units that have to elapse between a visit event at a component of type T_{C_1} and a visit event at a component of type T_{C_2} . Thus, if $w_T = 0$ and e_T is undirected, a visit event at a component of type T_{C_1} always has to occur simultaneously with a visit event at a component of type T_{C_2} (and vice versa).

If e_T is directed, it additionally determines that the visit events must occur in a specific order following the edge's orientation. For example, if $D_T = \{T_{C_2}\}$ then a visit event at a component of type T_{C_2} must always be preceded by a visit event at a component of type T_{C_1} . In this case w_T specifies the minimum number of time units that have to elapse between these visit events (if $w_T = 0$, the visit events may also occur simultaneously).

If a component type T_{C_0} has multiple undirected and/or incoming «time» edges $e_{T_i} = (\{T_{C_0}, T_{C_i}\}, D_{T_i}, L_i, w_{T_i})$ where $i \neq 0$ and $T_{C_i} \notin D_{T_i}$, it has to be differentiated if a component of type T_{C_0} is affected by one or more of these edges: All «time» edges e_{T_i} with $T_{C_0} \in L_i$ are connected to T_{C_0} with a logical AND relation and thereby must be applied at once if a visit event occurs at a component of type T_{C_0} . From all other «time» edges e_{T_i} with $T_{C_0} \notin L_i$ at least one must be applied if a visit event occurs at a component of type T_{C_0} .

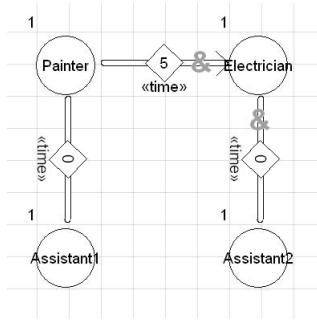


Fig. 5. Example of «time» edges: The painter's work has to precede the electrician's work (5 units of drying time). The assistants must always work simultaneously to their corresponding foreman. The ampersands determine that for every visit event of the electrician both time dependencies must be fulfilled. (Source: AbstractSwarm software tool, adapted from [2], p. 15).

Graphically, an AND relation is indicated by an ampersand between the rhombus and the connection point of a «time» edge to a component type.²

2.3 Attributes

Usually it is not sufficient to only model the existence of the components (agents and stations) that are part of a MAS. In this case, components have to be modeled in more detail by describing their properties as well. This can be achieved by assigning attributes to the component types.

AbstractSwarm offers ten generic attributes to define properties for component types. These attributes are designed to be generally applicable to a wide range of problem domains. To keep the resulting graphs as clear as possible, the assignment of all these attributes is optional. Thus, an intuitive default behavior is defined for all attributes that are not assigned to a component type. Thereby, every attribute that is assigned to a component type can be seen as a constraint for its default behavior.

An attribute p is defined as a tuple

$$p := (n, v) \quad (7)$$

where n is (an acronym of) the attribute name that describes the attribute and $v \in \mathbb{N} \setminus \{0\}$ is the attribute value.

Attributes are graphically represented by their names and their values noted at the top right corner of a component type.

Some attributes can be assigned to any kind of component type, others can be assigned to agent types or station types only.

Below, all attributes that can be assigned to any kind of component type are introduced at first. After that, the agent type and station type specific attributes are explained.

Component Type Attributes. Attributes that can be assigned to both agent and station types have an attribute name $n_C \in N_C$ with

$$N_C := \{\text{Prio.}, \text{Freq.}, \text{Nec.}, \text{Time}, \text{Cycle}\} . \quad (8)$$

The acronyms “Prio.,” “Freq.” and “Nec.” represent the names of the Priority, the Frequency and the Necessity attribute. “Time” and “Cycle” are the names of the Time and the Cycle attribute.

² As an extension, one might think of a number $k \in \mathbb{N}$ instead of an ampersand, which indicates an AND relation for only those «time» edges having the same k . In this way, more complex logical relations could be modeled. However, because of the transitivity of «time» edges, the syntax introduced here has been sufficient in all studied cases and is much simpler.

Priority. The Priority attribute models the urgency of visit events. If an agent type T_{A_1} has a Priority attribute $\hat{p}_{\text{Prio.}} = (\text{Prio.}, \hat{v}_{\text{Prio.}}) \in P_{A_1}$, agents of type T_{A_1} enforce their visits prior to agents of any other type T_{A_2} having a Priority attribute $p_{\text{Prio.}} = (\text{Prio.}, v_{\text{Prio.}}) \in P_{A_2}$ with $v_{\text{Prio.}} < \hat{v}_{\text{Prio.}}$ (regardless of any disadvantages). If a station type T_{S_1} has a Priority attribute $\hat{q}_{\text{Prio.}} = (\text{Prio.}, \hat{w}_{\text{Prio.}})$, stations of type T_{S_1} are preferably visited over stations of any other type T_{S_2} having a Priority attribute $q_{\text{Prio.}} = (\text{Prio.}, w_{\text{Prio.}})$ with $w_{\text{Prio.}} < \hat{w}_{\text{Prio.}}$. Components of types without Priority attribute count as least urgent.

Frequency. The Frequency attribute limits the total number of visit events. If an agent type T_A has a Frequency attribute $p_{\text{Freq.}} = (\text{Freq.}, v_{\text{Freq.}}) \in P_A$, every agent of type T_A must visit exactly $v_{\text{Freq.}}$ stations. If a station type T_S has a Frequency attribute $q_{\text{Freq.}} = (\text{Freq.}, w_{\text{Freq.}}) \in P_S$, every station of type T_S must be visited exactly $w_{\text{Freq.}}$ times. Components that have reached a number of visit events equal to the value of their Frequency attribute are no longer considered during a simulation.

Necessity. The Necessity attribute resembles the Frequency attribute: If an agent type T_A with «visit» edges $e_{V_i} = (\{T_A, T_{S_i}\}, b_i)$ has a Necessity attribute $p_{\text{Nec.}} = (\text{Nec.}, v_{\text{Nec.}}) \in P_A$, every agent of type T_A must visit every station of the types T_{S_i} exactly $v_{\text{Nec.}}$ times. If a station type T_S with «visit» edges $e_{V_j} = (\{T_{A_j}, T_S\}, b_j)$ has a Necessity attribute $q_{\text{Nec.}} = (\text{Nec.}, w_{\text{Nec.}}) \in P_S$, every station of type T_S must be visited exactly $w_{\text{Nec.}}$ times by every agent of the connected types T_{A_j} . Components of types without Necessity and Frequency attribute are not limited regarding the number of visit events.

Time. The Time attribute limits the duration of visits. If an agent type T_A has a Time attribute $p_{\text{Time}} = (\text{Time}, v_{\text{Time}}) \in P_A$, agents of type T_A visit stations for exactly v_{Time} time units. Analogously, if a station type T_S has a Time attribute $q_{\text{Time}} = (\text{Time}, w_{\text{Time}}) \in P_S$, stations of type T_S are visited by agents for exactly w_{Time} time units. If an agent $a \in A$ of type T_A visits a station $s \in S$ of type T_S with $v_{\text{Time}} \neq w_{\text{Time}}$, the duration of the visit is determined by $\min(v_{\text{Time}}, w_{\text{Time}})$. Components of types without Time attributes are not limited regarding the duration of their visits (thus, a visit would last until the end of a simulation).

Cycle. The Cycle attribute determines after how many visit events the condition of a directed «time» edge must be fulfilled again. If two component types T_{C_1} and T_{C_2} are connected by a «time» edge $e = (\{T_{C_1}, T_{C_2}\}, \{T_{C_2}\}, L, w_T)$ and T_{C_1} has a Cycle attribute $p_{\text{Cycle}} = (\text{Cycle}, v_{\text{Cycle}}) \in P_{C_1}$, a component of type T_{C_2} must have v_{Cycle} visit events after every visit event at a component of type T_{C_1} . Thus, the Cycle attribute has no effect if the component type has no outgoing «time» edges. If a component type has outgoing «time» edges but a Cycle attribute is not assigned, the components of the connected types can have an arbitrary number of visit events, once the conditions of the «time» edges have been fulfilled during a simulation.

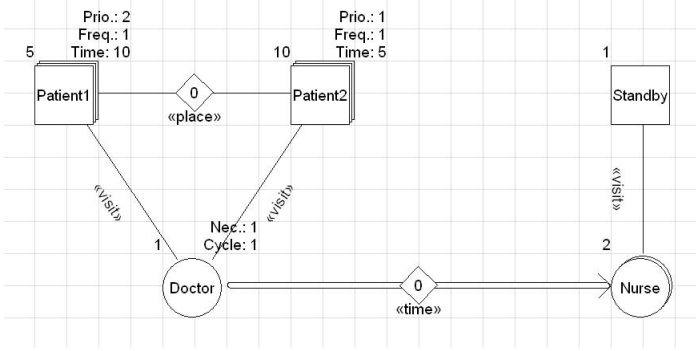


Fig. 6. Example of component type attributes: At a medical station, a doctor must treat every patient once (Nec.), the patient then no longer being considered by the simulation (Freq.). Patients of type 1 have a higher priority than patients of type 2 (Prio.), even if patients of type 2 can be treated in less time (Time). A nurse must be present for standby immediately after the doctor treated a patient. This must be the case every time the doctor treats a patient (Cycle). (Source: AbstractSwarm software tool, adapted from [2], p. 19).

Agent Type Specific Attributes. Attributes that can be assigned to agent types have an attribute name $n_A \in N_C \cup N_A$ with

$$N_A := \{ \text{Cap.}, \text{Size}, \text{Speed} \} . \tag{9}$$

The acronym “Cap.” represents the name of the Capacity attribute. “Size” and “Speed” are the names of the Size and the Speed attribute.

Capacity. Agents are able to transport items from one station to another: An item is loaded whenever a station is visited by an agent $a \in A$ of type T_A and the item is delivered to the next station that is visited by a . The Capacity attribute limits the available loading space for items. If T_A has a Capacity attribute $p_{\text{Cap.}} = (\text{Cap.}, v_{\text{Cap.}}) \in P_A$, the maximum loading space of every agent of type T_A is equal to $v_{\text{Cap.}}$. The item size can be defined by the Item attribute which can be assigned to station types (see section “Item”). Whenever an agent loads an item, its currently available loading space is decreased by the size of the loaded item. Whenever an agent delivers an item, its currently available loading space is increased again by the item size. Agents of agent types without Capacity attribute are not limited regarding their available space for items, thus they can load an arbitrary number of items of any size.

Size. The Size attribute limits the space needed by an agent when visiting a station. If an agent type T_A has a Size attribute $p_{\text{Size}} = (\text{Size}, v_{\text{Size}}) \in P_A$, the space needed by an agent $a \in A$ of type T_A when visiting a station is equal to v_{Size} . The total available space of a station $s \in S$ of type T_S can be defined by

the Space attribute which can be assigned to station types (see section “Space”). Whenever an agent visits a station, the station's currently available space is decreased by the agent size. Whenever an agent leaves a station, the station's currently available space is increased again by the agent size. If a station has not enough space left, the station cannot be visited. Agents of agent types without Size attribute are not limited regarding their size, thus by visiting a station, the station always becomes completely filled.

Speed. The Speed attribute limits the speed of agents when moving from one station to another along «place» edges. If an agent type T_A has a Speed attribute $p_{\text{Speed}} = (\text{Speed}, v_{\text{Speed}}) \in P_A$, agents of type T_A perform a movement of one distance unit on a «place» edge every v_{Speed} time units. Agents of types without Speed attribute are not limited regarding their speed, thus their speed is arbitrarily high and moving on «place» edges does not take any time.

Station Type Specific Attributes. Attributes that can be assigned to station types have an attribute name $n_S \in N_C \cup N_S$ with

$$N_S := \{\text{Item}, \text{Space}\} \quad (10)$$

where “Item” and “Space” are the names of the Item and the Space attribute.

Item. The Item attribute limits the size of items that can be transported by agents from one station to another. If a station type T_{S_0} has an Item attribute $p_{\text{Item}} = (\text{Item}, v_{\text{Item}}) \in P_{S_0}$, the size of the item loaded by an agent $a \in A$ of type T_A when visiting a station $s \in S_0$ of type T_{S_0} is equal to v_{Item} . The available loading space of a is defined by the Capacity attribute which can be assigned to its agent type T_A (see section “Capacity”). The delivery targets of items are determined by outgoing «place» edges $e_{P_i} = (\{T_{S_0}, T_{S_i}\}, \{T_{S_i}\}, w_{P_i})$ with $i \neq 0$. Thus, stations of types with outgoing «place» edges can be considered senders and stations of types with incoming «place» edges can be considered receivers. Stations of types with incoming and outgoing «place» edges can only send an item if at least one item was received via every incoming «place» edge (“transshipment nodes”). Item and Capacity attributes determine after how many visit events the condition of a «place» edge must be fulfilled again (similar to the Cycle attribute determining after how many visit events the condition of a «time» edge must be fulfilled again). Stations of station types without Item attribute have items that are not limited regarding their size, thus agents with any capacity become entirely loaded while visiting.

Space. The Space attribute limits the available space for visiting agents. If a station type T_S has a Space attribute $p_{\text{Space}} = (\text{Space}, v_{\text{Space}}) \in P_S$, the maximum space of every station $s \in S$ of type T_S is equal to v_{Space} . The agent size can be defined by the Size attribute which can be assigned to agent types. The mechanism between Size and Space attributes corresponds to the mechanism between Item and Capacity attributes (see section “Size”). Stations of station

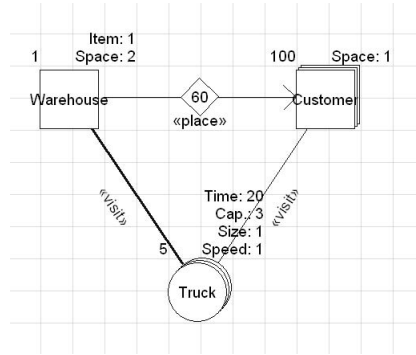


Fig. 7. Example of agent and station specific attributes: Trucks transport products from a warehouse to the customer. The transportation takes 60 time units (Speed). A truck can load three units at a time (Item, Cap.). Only two trucks can be loaded/unloaded simultaneously (Size, Space). (Source: AbstractSwarm software tool, adapted from [2], p. 23).

types without Space attribute are not limited regarding the available space for agents, thus they can be simultaneously visited by an arbitrary number of agents of any size.

2.4 Perspectives

One of the major difficulties when modeling MAS for extensive problems is the seemingly unmanageable complexity. In complex scenarios, there is often no obvious starting point from which a model could be constructed, since all entities and parameters of the problem domain seem to correlate.

AbstractSwarm offers the perspective syntax for the decomposition of complex problems. A perspective $\psi \in \Psi$ is defined as a tuple

$$\psi := (f, S_{T_C}) \quad (11)$$

where f is the perspective name (describing the facets of a subsystem represented by the perspective) and $S_{T_C} := S_{T_A} \cup S_{T_S}$ is a set of component types.

A perspective is graphically represented by a rectangle that encloses all agent and station types contained in S_{T_C} . The perspective name f is noted at the top left area of the rectangle. Perspectives represent parts of a MAS considering only certain facets of the underlying problem (e.g. subsystems with simplified objectives). The decomposition into perspectives is subjective, thus multiple different decompositions of the same problem may exist, depending on the aims of the MAS to be modeled.

Components may have various roles in different perspectives. This can be expressed by modeling multiple component types that represent the same components for each perspective. It can also be useful to represent components as stations in one perspective and as agents in another or to combine multiple

component types of one perspective to a single component type in another. The component types can then be synchronized with «time» edges to compose a complex MAS from the simplified perspectives.

Perspectives only serve as structural elements to decompose complex problems systematically and do not have any semantics for the simulation.

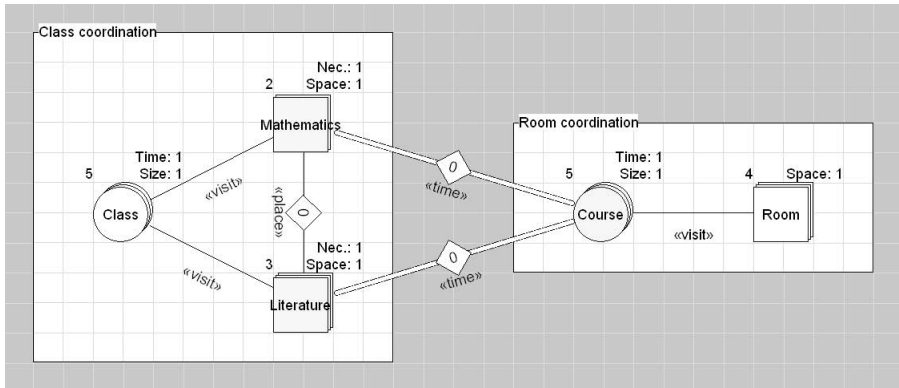


Fig. 8. Example of perspectives: A simple timetabling problem. Assignment of classes to courses (perspective “Class coordination”) and of courses to rooms (perspective “Room coordination”). (Source: AbstractSwarm software tool, adapted from [2], p. 25).

3 Conclusion and Future Work

The graphical modeling language introduced in this paper offers a common base for the simulation of problems as MASs. On the one hand, the simulation environment and agent behavior can be implemented without any expert knowledge of specific problem domains. On the other hand, it is possible to model problems without extensive knowledge of multi-agent technologies.

The simulation results can be easily obtained as Gantt charts by logging whenever an agent visits or leaves a station. Thereby, the results are comparable across problem domains and common evaluation criteria (e.g. the total waiting time of agents) can be defined.

Based on the results presented in this paper, a standard implementation of the simulation system will be developed as future work. Furthermore, the simulation of larger problems will be analyzed in a subsequent paper.

References

1. Abdelraouf, I., Abdennadher, S., Gervet, C.: A Visual Entity-Relationship Model for Constraint-Based University Timetabling. In: Abreu, S., Oetsch, J., Pührer, J., Seipel, D., Tompits, H., Umeda, M., Wolf, A. (eds.) 19th International Conference on Applications of Declarative Programming and Knowledge Management, pp. 183–194. TU Wien, Wien (2011)

2. Apeldoorn, D.: Entwicklung eines abstrakten Modells zur Modellierung von Logistikproblemen. Johannes Gutenberg-Universität Mainz, Mainz (2006)
3. Bloech, J., Ihde, G.B. (eds.): Vahlens Großes Logistik Lexikon. C.H. Beck & Franz Vahlen, München (1997)
4. Henriksen, J.O.: SLX: The X is for Extensibility. In: Joines, J.A., Barton, R.R., Kang, K., Fishwick, P.A. (eds.) Proceedings of the 2000 Winter Simulation Conference, pp. 183–190. IEEE Press, New York (2000)
5. Klaus, P., Krieger, W. (eds.): Gabler Lexikon Logistik: Management logistischer Netzwerke und Flüsse. Gabler, Wiesbaden (2004)
6. Klügl, F.: SeSAM: Visual Programming and Participatory Simulation for Agent-Based Models. In: Weyns, H., Uhrmacher, A. (eds.) Multi-Agent Systems: Simulation and Applications, pp. 477–508. CRC Press, Boca Raton (2009)
7. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: MASON: A Multi-Agent Simulation Environment. *Simulation: Transactions of the Society for Modeling and Simulation International* 81(7), 517–527 (2005)
8. Pfohl, H.-C.: Logistikprobleme: Betriebswirtschaftliche Grundlagen. Springer, Berlin (2000)
9. Pinedo, M.: Scheduling: Theory, Algorithms, and Systems. Springer, New York (2012)

A Principled Approach for Smart Microgrids Simulation Using MAS

Gillian Basso, Vincent Hilaire, Fabrice Lauri, Damien Paire,
and Arnaud Gaillard

IRTES-SeT, UTBM
90010 Belfort Cedex, France
gillian.basso@utbm.fr
www.multiagent.fr

Abstract. Energy management is, nowadays, a subject of uttermost importance. Indeed, we are facing growing concerns such as petroleum reserve depletion, earth global warming or power quality (e.g. avoiding blackouts during peak times). Smartgrids (SG) is an attempt to solve such problems, by adding to power grids bi-directionnal communications and ICT capabilities in order to provide an intelligent autonomic management for the grid. Microgrids are a possible implementation of SG. They are defined by rather small power systems, composed of power sources (some may be renewables) and loads, that can be connected or not to the main grid. In this context, simulation is an appropriate approach for studying the introduction of SG in existing systems. Indeed, it avoids the deployment of real, costly infrastructures and reduce experimental risks. This paper presents a microgrid simulator based on Multi-Agent Systems (MAS).

Keywords: multi-agent based simulation, organisational methodology, smart grids, microgrid.

1 Introduction

The management of an electrical grid is a major challenge of the 21st century. The existing electrical grid has to evolve to include new distributed generators like microturbines, photovoltaic panels, fuel cells and wind turbines, and also to balance the petroleum and carbon reserve depletion. In [13], authors detail problems emphasized by the grid evolution, such as demand-side management, electrical vehicles adoption or new user types (and behavior) like energy prosumers. Smartgrids is an attempt to solve such problems, by adding to power grids bi-directionnal communications and ICT capabilities in order to provide an intelligent autonomic management for the grid.

However, deploying an intelligence over the current monolithical grid is a hard challenge. A possible solution consists in decomposing the grid into smaller interconnected power grids called microgrids. Microgrids [10] can be defined as low-voltage parts of the energy network that comprise loads, decentralized

sources (maybe renewables) and local storage systems. Microgrids can operate either connected to the main distribution grid (and thus to others microgrids), or disconnected. In the latter case, called islanded, microgrids are in charge of controlling their power balance and voltage stability in a local way.

There are several existing approaches that try to introduce intelligence within microgrids [12,3]. These approaches differ both with the technical characteristics of the microgrids and with the paradigm chosen for the intelligent management. It is thus difficult to compare these approaches or generalize them. The contribution of this paper consists in proposing a multi-agent based simulator for microgrids that is able to (i) reliably simulate the grid operation with a broad range of devices (ii) prototyping intelligent control mechanisms over these grids.

The use of a simulator in such a context avoids the design of new infrastructures and reduces significant investments. Nowadays, different power system simulators are developed to offer a vision of the future power systems. In the sequel of this section, the most advanced and used simulators are described.

EPRI's OpenDSS [4], a Distribution System Simulator is a very flexible and expandable research platform which wants to be a foundation for new distribution analysis applications particularly for distributed generators. It supports frequency domain analysis commonly performed on electric utility power distribution systems and provides a large collection of renewables models.

GridLAB-D[1] is a power distribution system simulation environment that provides to users tools for data analysis such as an agent-based and information-based modelling and validation for rate structures and consumer reaction. The core of GridLAB-D has an advanced algorithm that simultaneously coordinates the state of millions of independent devices, each of which is described by multiple differential equations, etc.

PowerWorld Simulator is a popular simulation software used to simulate high voltage power systems. Using this tool, it is possible to perform power flow analysis on a system with up to 100,000 buses. PowerWorld offers several add-ons such as voltage and power stability analysis, modeling and evaluation of geomagnetic disturbances, etc.

We also want to mention the MATLAB/Simulink tool **SimPowerSystems** that provides component libraries and analysis tools for modelling and simulating electrical power systems. The libraries include models of electrical power components, including three-phase machines, electric drives, and components for applications such as flexible AC transmission systems and renewable energy systems.

These simulators are very useful for data analysis but are not developed to easily and dynamically integrate intelligences and especially distributed intelligences.

The integration of intelligences is the main objective of our simulator. We want to offer to users a virtual and multi-agent environment for their artificial intelligences which aim to be integrated in a real power system in the everyday life.

The section 2 presents the multi-agent based simulation of microgrids. In this section, a comparison of microgrids and multi-agent system is firstly provided, then the organizational approach of our simulator is presented. A description of how the time is handled within our simulator is also given and also some device model implementations. The section 3 presents the first results obtained with our simulator and gives a comparison of these results obtained by `SimPowerSystems` tool for a given scenario.

2 Simulation of Microgrids

To explain a simulation of microgrid, the concepts that the system has to implement is developed. A microgrid is a set of entities. An entity can be defined as an energy container. It may be a device that provides energy. This energy can be positive for a producer or negative for a consumer. An entity can also be a transmitter that transfers energy between other entities. In our simulator, the device class can be divided into 3 subclasses:

- loads that consume energy, assuming that the energy consumed is considered as negative,
- sources that produce energy, assuming that the energy produced is considered as positive, and
- storage systems that can consume or produce energy depending on its action (charging or discharging).

The transmitter class can be divided into 2 subclasses:

- the converter that links devices to other entities, and must transform the energy to match the need of devices
- the bus that is a link between converters or devices.

In order to connect an intelligent management to our simulator, two concepts are developed: the sensors that offer data access from entity to the outside of the simulator and the actuators that receive information from the outside to influence the behaviour of entities.

We also define constraints based on physical laws that the simulator continuously verifies while running a scenario:

- each device has its own power dynamic, depending on its internal characteristics, these characteristics are fixed and could be defined by users,
- during all the simulation, the Kirchhoff first law shall always be true: the sum of currents flowing into a node is equal to the sum flowing out of that node,
- the voltage of the grid depends on the power flow between the different elements, the relation is detailed in the section 2.4.

These physical laws create a link between the current delivered by each device plugged in the grid and the voltage stability of the grid.

Considering the purpose of our simulator, the concepts defined above and the physical laws that the simulator has to take into account, the developed capabilities offered to users are:

- a set of initial conditions, to define the characteristics of the entities and the state of the simulation at the beginning of the simulation at time $t = 0$,
- a real time simulation, thanks to the time management (see 2.3),
- an open system, with dynamic actions on microgrids (for example, adding or removing devices during a simulation) or on the devices which can be controlled by external intelligences,
- a monitoring, with extraction of information of devices and grid. Currently, the observables are the current of each device, the voltage of the transmitters and state of charge for storage systems.

2.1 Microgrid Simulation and Multi-agent System

The simulator presented in this paper does not provide a global intelligent system but aims to be a multi-agent and multi-level environment that offer to whatever intelligence different scenarii in which they can operate. The separation of the physical implementation (or the environment) and the intelligence is important to design an intelligent system. In [7], four main aspects are described :

- the agent behaviour** corresponds to the intelligence of the network and is out of the scope of this paper,
- the environment** corresponds to the core of the simulator,
- the scheduling** corresponds to the time management,
- the interactions** corresponds to the influence of the intelligence agents on the environment (the simulator).

These aspects have to interact together but must be independent in order to clearly separate the models from their implementations, and the intelligent system from its environment.

The simulator presented in this paper is developed with JANUS [8] a multi-agent platform developed with JAVATM. This platform offers organisational model, network peer-to-peer and holonic systems.

These different characteristics respectively help to simulate the dynamic variation of a grid, to distribute the simulator easily and to develop hierarchical (or holistic) microgrids.

A powergrid is composed of several devices geographically distant from each other. These devices can be connected or disconnected at any time and continuously change their consumptions or productions. The properties of an actor of a powergrid can be put in parallel with the properties of an agent [14] in a multi-agent system:

- autonomy:** a device cannot directly modify the consumption/production of another device,

- social ability:** every device presented in a powergrid have to exchange energy with other devices,
- reactivity:** the devices receive energy but can also observe the powergrid to dynamically change its own consumption/production,
- pro-activeness:** a device can change its internal state without considering the state of the powergrid.

2.2 Organizational Approach

As previously expressed, this simulator is based on JANUS, an organization based multi-agent platform. Indeed, JANUS is built upon the ASPECS metamodel [2] in which the concepts of role and organization are first-class entities. An agent is an autonomous entity that has specific individual goals and the intrinsic ability to realize some capacities playing different roles. A role is an expected behaviour and a set of rights and obligations in the organization context. The goal of each role is to contribute to the fulfilment of (a part of) the requirements of the organization within which it is defined. An organization is defined by a collection of roles that take part in systematic institutionalized patterns of interactions with other roles in a common context. This context consists in shared knowledge and social rules/norms, social feelings, etc. and is defined according to an ontology. The aim of an organization is to fulfil some requirements.

In JANUS, the organization is implemented as a first-class entity (a class in the object-oriented sense), which includes a set of role classes. An organization can be instantiated in the form of groups. Each group contains a set of instances of different classes of roles associated with the organization which it implements. A role is local to a group, and provides agents playing the role and the means to communicate with other group members. One of the most interesting aspects of JANUS covers the implementation of roles as first class entity. A role is seen as a full-fledged class, and the roles are implemented independently of the entities that play them.

The organizational approach of the simulator is divided into three aspects:

- the microgrid organization** managing the exchange of energy between all the entities, this is the core of the simulator and represents the environment in an intelligent system.
- the hierarchical organization** allowing the use of microgrids as a device,
- the communication organization** required to communicate with the outside.

The microgrid organization (see figure 1) is composed of 4 roles: the device role, the transmitter role, the regulator role and the time manager role.

First, the device role has to be played by any entity that provides energy to the grid. The provided energy is positive for a producer, for example, a photovoltaic panel, or negative for all device that consumes energy, it could be a home, a microwave, etc.. A storage system, a battery for example, also has to play a device role. It can be a producer when delivering energy or a consumer when charging.

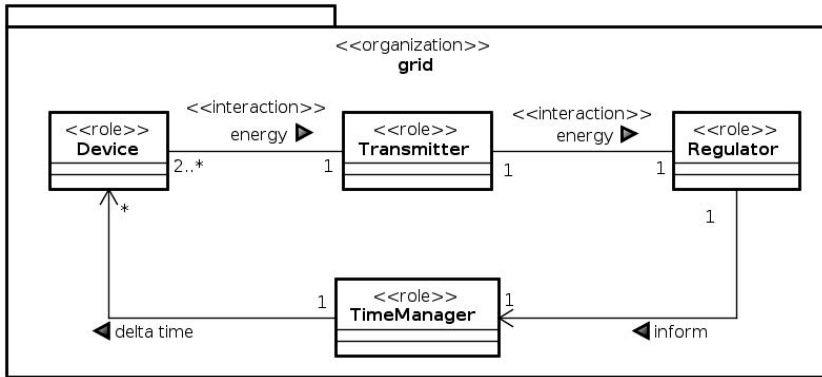


Fig. 1. The CRIO diagram of the microgrid organization

The second role, the transmitter role has to be played by every entity that will transfer energy between two or more roles and do not provides energy by itself. This role also offers the possibility to simulate losses during the transfer. An electrical bus is a typical example of an entity which has to play this type of role. A converter is another entity that can play the transmitter role to transfer and convert energy to entities with different characteristics.

The regulator role is connected to a transmitter to control the power flow through it. The regulation includes the stability of the network, it could be a frequency stability and/or a voltage stability depending on the entity playing this role. This role can also break the energy transmission if it considers the network as unstable.

The time manager guarantees that the time of the simulation is consistent. All the roles played in an organization are synchronized following the time given by the time manager (see section 2.3). Each cycle of a simulation is started when the time manager role decide to increment the time of the simulation.

The hierarchical or multi-levels aspect of grids is modelled by using two organizations (see figure 2).

The microgrid organization is linked to an upper-level organization. The principle is based on the connectivity of the electrical entities. Each microgrid, if not islanded, can be connected to other microgrids or more important grids by means of devices represented by a connector role.

The zone role represents a part of a network that can be independently simulated and have to be connected to another part of the network also independently simulated represented by another Zone role. These two roles work together (i.e. to exchange energy) has to be linked by a connector role.

The connector role ensures the communication of two or more zone roles. As the zone roles can simulate parts of network by different way, the connector role has to receive the energy sent by all the zone roles and transform the energy to ensure that the energy is valid for the other zone role. It roughly works as the transmitter role works in the microgrid organization.

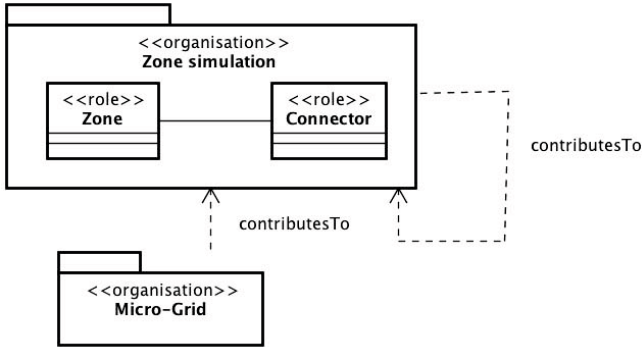


Fig. 2. The CRIO diagram of the hierarchical organization

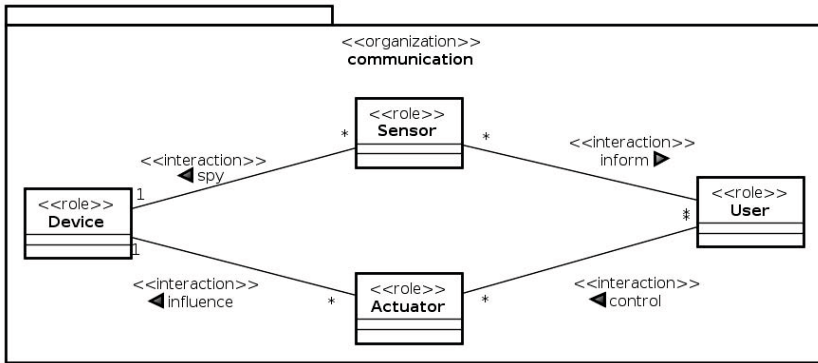


Fig. 3. The CRIO diagram of the communication organization

Eventually, the communication organization (see figure 3) represents the communication between the intelligence of system and the environment in which it operates. It is composed of 4 roles: the device role, the sensor role, the actuator role and the user role.

Firstly, the device role offers a two-way communication. The agent that played this role can change its behaviour following information received from the outside and can also be monitored by extracting local information.

Secondly, the sensor role is played by an agent that want to monitor a device role. The communication between a sensor role and a device role is a one-way communication, the agent cannot influence the device role by sending data.

Thirdly, the actuator role is played by an agent that has to send data to a device role. This one-way communication offer to the device role a aperture to the outside. The sent data shall not change the behaviour of the device role directly but must inform the device role of any change in its environment.

Eventually, the user role is a boundary role that can be played by any external intelligence and must be linked to some sensor roles and actuator roles to be aware of information of the simulation and influence the device of the latter.

For example, an intelligence wants to change the production of a fuel cell following the consumption of an engine. To do so, it has to play two user roles: one to monitor the consumption of the engine, and one to influence the production of the fuel cell.

2.3 Time Management

The time management represents the scheduling of the environment in a intelligent system. When simulating MASs, time management is a critical point because the timing delays introduced by the underlying execution platform may affect the simulation results [9]. Following the classification in [5], the time management used by our simulator can be classified as conservative with a Lower Bound on the Time Stamp. These approaches do not allow an agent to process an event until it has been guaranteed to be safe. In this algorithm, all messages sent during the same time stamp are safe. No future or past messages will be processed during all the simulation. Also, if the time is incremented following the real time, the simulator implements as real as possible the time flow and the exchange of messages.

In our simulator, an event becomes safe at the end of the time frame in which it was launched. Thus the amount of energy provided by the devices during a time frame will be managed by the regulator at the end of a time frame. It ensures that the time frame is the time unit of our simulator. As the duration of the time frame of the simulator can be set by the user the simulation speed can be controlled by user. The simulator can also be executed as fast as allowed by the underlying hardware.

The implementation is realized by the time manager role of the microgrid organization. After each step of the simulation, the time manager informs the device roles that they have to send power to the transmitter. The transmitter ensures the power flow between each device connected to itself. If connected device roles do not provide the energy during the time frame the transmitter assumes that no energy transfer is made. Then it sends the amount of power to the regulator to check the stability of the system. The time manager finally increments the time of the simulation.

2.4 Physical Model of Devices

Currently, the simulator contains several types of devices which can be producer, consumer or storage systems:

- photovoltaic panels and wind turbines whose energy production is directed by weather conditions. They are stochastic producers,
- fuel cells whose power can vary between 0 and a maximal power following their own dynamics.

- storage systems which have their own dynamic and their own capacity. Both of these properties are limited depending on the type of storage system. Batteries have a low dynamic but a great capacity compare to supercapacitors which have a high dynamic but a small capacity),
- electric network is an equivalent to the existing network in a microgrid: no limitations are made in its properties.
- a set of loads (consumers). Currently these loads are not controllable.
- a DC bus allows other devices to exchange energy. Its voltage shall be maintained in a defined voltage range otherwise the system will crash.

The developed models detailed below are simple and will be completed when a validation with real model will be realized. For example the electrical impedance is negligible and the converters are considered as gains.

Renewables. As the renewable energy resources are currently stochastic, they are simply represented as current profiles. These data can be read from a file or a stream.

Fuel Cells - Simplified Model. The power of the fuel cell is calculated according to the current i_{fc}^{dc} (the current coming from the fuel cell on the DC bus) following this relation:

$$P_{fc} = \frac{i_{fc}^{dc} \cdot v_{dc}}{\eta_{fc}}$$

with η_{fc} the converter efficiency (between 0 and 1) linked to the fuel cell.

The instant consumption of hydrogen in terms of electrical power of the fuel cell is given by:

$$q_{H_2} = \frac{P_{fc}}{\eta_{fc}(P_{fc}) \cdot HHV_{H_2}}$$

with q_{H_2} is the hydrogen output. $\eta_{fc}(P_{fc})$ is the efficiency (between 0 and 1) of the fuel cell depending on the functional point. HHV_{H_2} is the Higher Heating Value of the hydrogen and is a constant at $140MJ/kg$ [6].

The quantity of hydrogen consumed during a interval Δt is:

$$m_{H_2} = \int_t^{t+\Delta t} q_{H_2} dt$$

The power of the fuel cell must be between 0 and the maximal power P_{fc}^{max} authorized by the system. The dynamic of the power also must be between the minimal dynamic power dP_{fc}^{min} (a negative value) and the maximal dynamic power dP_{fc}^{max} (a positive value). Finally, when the fuel cell cannot have hydrogen the production of energy will be null.

Storage Systems - Simplified Model. The storage systems are modelled by an energy balance. The power P_{ss} of the system can be calculate by :

$$P_{ss} = \begin{cases} \frac{i_{ss}^{dc} \cdot v_{dc}}{\eta_{ss}} & \text{if } i_{ss}^{dc} \geq 0 \text{ (discharging)} \\ i_{ss}^{dc} \cdot v_{dc} \cdot \eta_{ss} & \text{if } i_{ss}^{dc} < 0 \text{ (charging)} \end{cases}$$

with η_{ss} is the efficiency (between 0 and 1) of the storage system linked to its converter.

The state of charge $SOC_{ss}(t)$ of the storage system is calculate following this relation:

$$SOC_{ss}(t) = SOC_{ss}^{init} - \frac{\int_0^t P_{ss} dt}{E_{ss}^{tot}}$$

with SOC_{ss}^{init} the initial state of charge of the storage system. E_{ss}^{tot} is the energy capacity of the system. The state of charge of a storage system also should be between minimal value SOC_{ss}^{min} and maximal value SOC_{ss}^{max} in order to preserve the system.

The power of storage system must be between the minimal power P_{ss}^{min} and the maximal power P_{ss}^{max} authorized by the system. Both of these value must be positive. The dynamic of the power also must be between the minimal dynamic power dP_{ss}^{min} (a negative value) and the maximal dynamic power dP_{ss}^{max} (a positive value). Obviously, if the storage system is empty, it cannot deliver energy, also if the storage system is full, it cannot receive energy.

Electric Grid. The electric network is the usual network already existing plugged with a microgrid. It can deliver or receive any energy of the microgrid.

Loads. As the loads are currently stochastic, they are simply represented as profiles of current from experiments. These data can be read from a file or a stream.

DC Bus - Simplified Model. The DC Bus is composed by a capacitor which required the voltage of the DC Bus. The voltage of the DC Bus is given by:

$$i_c = -C \frac{dv_{dc}}{dt}$$

with i_c is the current in the capacitor, C its capacity and v_{dc} the voltage across the capacitor. The current i_c is inferred from the current balance on the DC Bus from the Kirchhoff law:

$$\sum_k i_k + i_c = 0$$

with k the number of devices plugged on the DC Bus and i_k the current deliver by the device k . Thereby the DC Bus voltage can be inferred from the following relation:

$$v_{dc} = v_{dc}^{init} - \frac{1}{C} \int i_c dt$$

with v_{dc}^{init} the initial voltage of the capacitor.

3 Experiments and First Results

3.1 Scenario

The validity of our simulator was empirically validated using a simple scenario and compared with a simulator developed thanks to **SimPowerSystems**.

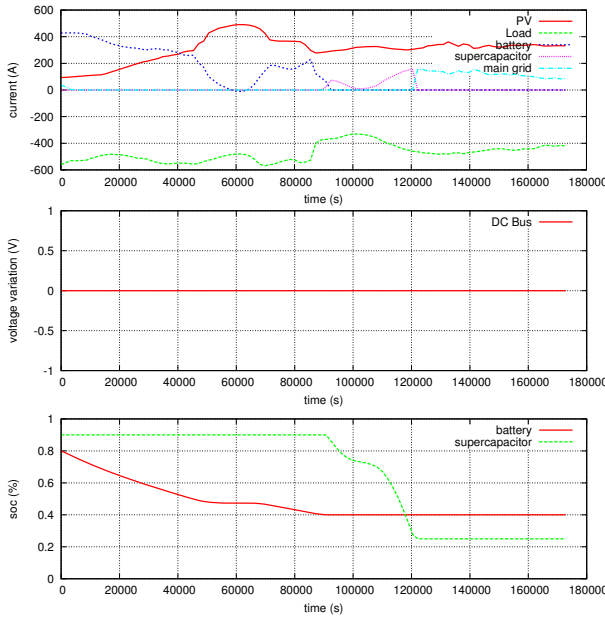


Fig. 4. Simulator: Top chart, current - Middle chart, voltage variation - Bottom chart, state of charge

In this scenario (see Figure 6), the microgrid is composed of a DC Bus connected to some renewables (a mix of photovoltaic and wind turbines), loads (a consumption equivalent to 50 homes), a fast storage system – a set of supercapacitors – with strong dynamic but low energy, a slow storage system – a set of batteries – with low dynamic but a lot of energy and the electrical grid assumed to have infinite dynamic and energy. The virtual duration of the simulation has been set to two days. The table 1 sums up the values of the data presented in the section 2.4. In this example, the time step between each stability verification is one second.

3.2 Artificial Intelligence

The intelligence developed to verify the validity of the simulator is a simple intelligence. The main objective of the intelligence is to keep the DC Bus voltage stable.

The intelligence first manages the behavior of the battery by regulating the energy. If the battery has not enough dynamic to regulate the DC Bus voltage

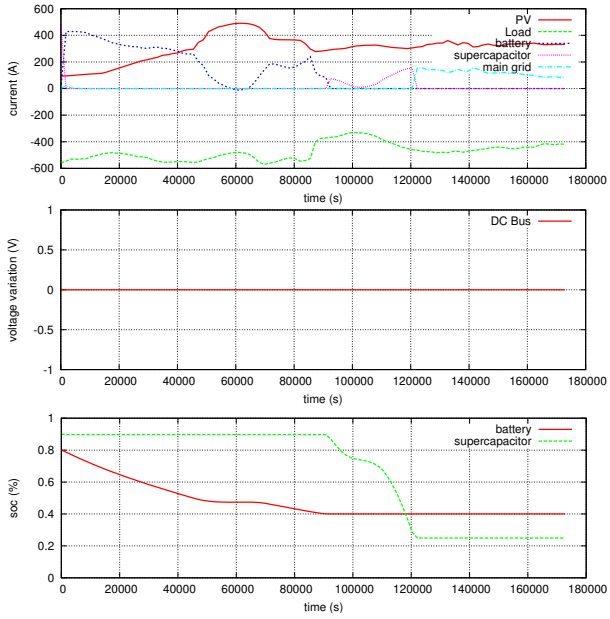


Fig. 5. SimPowerSystems: Top chart, current - Middle chart, voltage variation - Bottom chart, state of charge

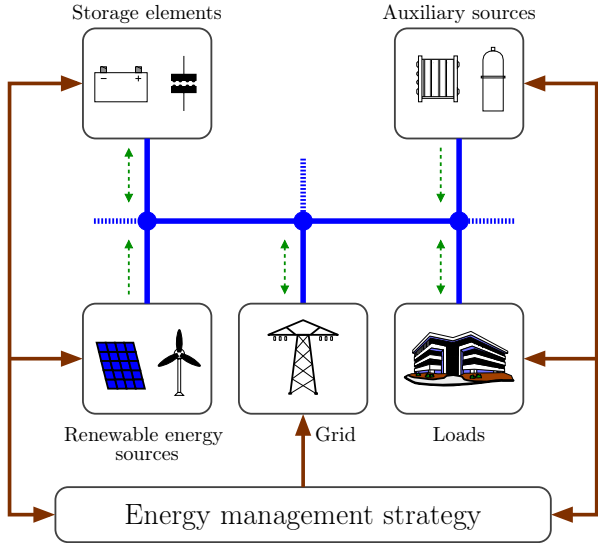


Fig. 6. A simple schema of the microgrid used for validation

Table 1. Data of the testing scenario

Renewable	
see Figure 4, top chart, curve in red	
Loads	
see Figure 4, top chart, curve in green	
Battery	
$SOC_{bat}^{min} = 0.25$	$SOC_{bat}^{max} = 0.9$
$P_{bat}^{min} = -300kW$	$P_{bat}^{min} = 300kW$
$\frac{dP_{bat}^{min}}{dt} = -2kW/s$	$P_{bat}^{min} = 2kW/s$
$E_{bat}^{tot} = 2MWh = 5kAh$	$\eta_{bat} = 1$
Supercapacitor	
$SOC_{sc}^{min} = 0.8$	$SOC_{sc}^{max} = 0.4$
$P_{sc}^{min} = -600 kW$	$P_{sc}^{min} = 600 kW$
$\frac{dP_{sc}^{min}}{dt} = -10kW/s$	$P_{sc}^{min} = 10 kW/s$
$E_{sc}^{tot} = 100 kWh = 250Ah$	$\eta_{sc} = 1$
DC Bus	
$v_{dc}^{mit} = 700V$	$C = 20mF$
$v_{dc}^{min} = 680V$	$v_{dc}^{max} = 720V$

or has a state of charge not correlated to the decision, then the supercapacitor comes into play and delivers some current according to a behavior similar to the battery’s one. In any case, the electrical network stabilizes the DC Bus voltage.

3.3 Results

The figure 4 shows the results obtained by our simulator and the figure 5 shows the results provided by the **SimPowerSystems** tool. In both figures, the top chart shows the variation of current of the devices. The second (middle) chart shows the success to keep the DC Bus voltage stable. The bottom chart shows the dynamic of both storage systems with their states of charge.

The comparison of the two figures highlights that the devices of our simulator globally have the same behaviors than those of the other simulator. To details the difference between the two results which is not visible on the charts, A numerical comparison is given the detail the variation of values between the two simulators:

- For the batteries, the mean of the differences between the intensities is 0.24A and the deviation is 5.74A.
- For the supercapacitors, the mean of the differences between the intensities is 0.29A and the deviation is 5.88A.
- For the electrical grid, the mean of the differences between the intensities is 0.30A and the deviation is 7.57A.

4 Conclusions and Future Works

In this paper a subject of importance for the Smart Grids community is presented. The proposition consists of a simulator capable of reproducing the behaviours of electrical devices in order to test MAS for Smart Grids management. This simulator is compared to MATLAB/Simulink `SimPowerSystems` which is, in the energy community, a well-known and widely used simulator. The results of this experiment shows that the outputs of the MAS are very close to `SimPowerSystems`. However, the architecture of the two simulators are radically different. `SimPowerSystems` is based on a centralized architecture and closeness is among it basic hypothesis. The MAS based simulator presented, on the contrary, is based on a pure agent architecture that allows distribution and openness.

Moreover, the presented principles allow to replace the simulator with real devices without changing the Smart Grids management part. Indeed, one of the aim of this simulator is to enable a prototyping approach for MAS dedicated to Smart Grids management. The simulator has been designed according to a specific MAS methodology, namely ASPECS [2]. This methodology is based on organizational concepts and is supported by a development platform that eases the implementation of the methodology concepts, namely Janus [8].

This new simulator is a first step toward the development of a library of devices to create a huge variety of scenarii. The idea is to enable the test of smart management strategies for power systems. One can thus test several techniques over a set of benchmarks for learning, or only to experience new dynamic controls. Currently the simulator contains some device implementations and is base on an DC bus architecture.

In the future, we plan to evaluate different MAS for Smart Grids in order to identify efficient approaches. These MAS will be tested within the proposed simulator and on real grids. More specifically, a reinforcement learning approach is under study [11] and an implementation of an AC bus architecture is planed.

References

1. Chassin, D.P., Schneider, K., Gerkenmeyer, C.: Gridlab-d: An open-source power systems modeling and simulation environment. In: IEEE/PES Transmission and Distribution Conference and Exposition, T&D, pp. 1–5 (April 2008)
2. Cossentino, M., Gaud, N., Hilaire, V., Galland, S., Koukam, A.: Aspecs: an agent-oriented software process for engineering complex systems. *Autonomous Agents and Multi-Agent Systems* 20, 260–304 (2010)
3. Dimeas, A.L., Hatzigiorgiou, N.D.: Agent based control for microgrids. In: IEEE Power Engineering Society General Meeting, pp. 1–5 (2007)
4. Dugan, R.: Open source software for simulating active distribution systems. *Active Distribution Management Tutorial* (2009)
5. Fujimoto, R.M.: Parallel simulation: distributed simulation systems. In: Winter Simulation Conference, pp. 124–134 (2003)
6. Gao, F., Blunier, B., Miraoui, A.: Proton Exchange Membrane Fuel Cells Modeling. Wiley-ISTE (February 2012)

7. Gaud, N., Galland, S., Gechter, F., Hilaire, V., Koukam, A.: Holonic multilevel simulation of complex systems: Application to real-time pedestrians simulation in virtual urban environment. *Simulation Modelling Practice and Theory* 16(10), 1659–1676 (2008)
8. Gaud, N., Galland, S., Hilaire, V., Koukam, A.: An organisational platform for holonic and multiagent systems. In: Hindriks, K.V., Pokahr, A., Sardina, S. (eds.) *ProMAS 2008*. LNCS, vol. 5442, pp. 104–119. Springer, Heidelberg (2009)
9. Helleboogh, A., Holvoet, T., Weyns, D., Berbers, Y.: Extending time management support for multi-agent systems. In: Davidsson, P., Logan, B., Takadama, K. (eds.) *MABS 2004*. LNCS (LNAI), vol. 3415, pp. 37–48. Springer, Heidelberg (2005)
10. Lasseter, R.H., Paigi, P.: Microgrid: a conceptual solution. In: 2004 IEEE 35th Annual Power Electronics Specialists Conference, PESC 2004, vol. 6, pp. 4285–4290 (June 2004)
11. Lauri, F., Basso, G., Zhu, J., Roche, R., Hilaire, V., Koukam, A.: Managing power flows in microgrids using multi-agent reinforcement learning. In: *Agent Technologies for Energy Systems, ATES* (May 2013)
12. Piagi, P., Lasseter, R.H.: Autonomous control of microgrids. In: *IEEE Power Engineering Society General Meeting*, page 8 (2006)
13. Ramchurn, S., Vytelingum, P., Rogers, A., Jennings, N.: Putting the “smarts” into the smart grid: A grand challenge for artificial intelligence. *Communications of the ACM* (2011)
14. Wooldridge, M., Jennings, N.R.: Intelligent agents: theory and practice. *The Knowledge Engineering Review* 10(02), 115–152 (1995)

Estimating Shapley Values for Fair Profit Distribution in Power Planning Smart Grid Coalitions

Jörg Bremer* and Michael Sonnenschein

University of Oldenburg, Environmental Informatics,
Uhlhornsweg 84, 26129 Oldenburg, Germany
{joerg.bremer,michael.sonnenschein}@uni-oldenburg.de

Abstract. In future, highly dynamic energy grids a likely scenario is to have dynamically founded groups of distributed energy resources that are in charge of jointly delivering a demanded load schedule for a certain time horizon. In market based scenarios, such a demanded load schedule would be a (day ahead) product that is to be delivered by a coalition of energy resources. Computational aspects of the underlying optimization problem or of proper coalition formation are already subject to many research efforts. In this paper, we focus on the question of fairly sharing the profit among the members of such a coalition. Distributing the surplus merely based on the absolute (load) contribution does not take into account that smaller units maybe provide the means for fine grained control as they are able to modify their load on a smaller scale. Shapley values provide a concept for the decision on how the generated total surplus of an agent coalition should be spread. In this paper, we propose a scheme for efficiently estimating computationally intractable Shapley values as a prospective base for future surplus distribution schemes for smart grid coalitions and discuss some first ideas on how to use them for smart grid active power product coalitions.

Keywords: Shapley value, active power load planning, smart grid, multi-agent system.

1 Introduction

In order to allow for a transition of the current central market and network structure of today's electricity grid to a decentralized smart grid, an efficient management of numerous distributed energy resources (DER) will become more and more indispensable.

We here consider rather small, distributed electricity producers that are supposed to pool together with likewise distributed electricity consumers and

* The Lower Saxony research network 'Smart Nord' acknowledges the support of the Lower Saxony Ministry of Science and Culture through the Niedersächsisches Vorab grant programme (grant ZN 2764).

prosumers (like batteries) in order to jointly gain more degrees of freedom in choosing load schedules. In this way, they become a controllable entity with sufficient market power.

Future smart grid scenarios are assumed to comprise the individual trade of active power products on markets by self-organized coalitions of pro-actively scheduled energy units [1]. In contrast to conventional, static virtual power plant (VPP) approaches with a rather fixed structure (group of units) operated by a single owner or operating company, we assume for the future dynamically (frequently) self-organized coalitions that cooperate on a short term but are operated by different owners. In scenarios with self-organized and dynamic formation of coalitions, units from different owners will group together and offer (among others) active power schedules or ancillary services to a market. Such a scenario differs from that of virtual power plants (if owned and operated by a single company), because of the economically cooperative behaviour: gains have to be distributed to different operators.

For this reason, each time a active power product is sold on the market, the jointly earned profit after delivery has to be distributed among all participants in a way that satisfies their owners but also reflects the capability of each single unit and considers their values for the coalition in a fair way. If a coalition of energy units has been assigned a product schedule, the coalition has to internally coordinate its members to jointly adduce the product schedule. In order to manage such a pool of DER, the following distributed optimization problem has to be solved: A partition of a demanded aggregate schedule has to be determined in order to fairly distribute the load among all participating units. Optimality usually refers to local (individual cost) as well as to global (e.g. environmental impact) objectives in addition to the main goal: Resemble the wanted overall load schedule as close as possible.

When determining an optimal partition of the schedule for load distribution, exactly one alternative schedule is taken from each unit's search space of individual operable schedules in order to assemble the desired aggregate schedule. For optimization, a scheduling algorithm (whether centralized or not) must know for each unit which schedules are operable and which are not. Therefore, the set of alternative, operable schedules (obeying multiple constraints like allowed power or voltage bands or buffer charging levels) has to be encoded by an appropriate, standardizable model for inclusion into optimization. An example for such a model has been presented by [2].

The rest of the paper is organized as follows: We start with some background on coalition games, especially in the context of load planning within the smart grid and briefly revisit the concept of Shapley values. We propose an estimation scheme combining a Monte Carlo approximation by [3] and a distributed algorithm for approximating individual coalition abilities regarding the achievement of a given load product. We present some simulation results and discuss possible future applications.

2 Related Work

2.1 Load Planning

Within the framework of today's (centralized) operation planning for power stations, different heuristics are harnessed. Examples from the research sector are for instance shown in [4] or in [5]. This task of (short-term) scheduling of different generators is often referred to as unit commitment problem and assigns (in its classical interpretation) discrete-time-varying production levels to energy generators for a given planning horizon [6]. It is known to be an NP-hard problem [7]. Determining an exact global optimum is, in any case, not possible in practice until *ex post* due to uncertainties and forecast errors. In practice the software package BoFIT is often used, harnessing a mixed integer model with operational constraints as an integral part of the implementation of the model [8]. This fact makes it hard to exchange operational constraints in case of a changed setting (e.g. a new composition of energy resources) of the generation system.

Coordinating a pool of distributed generators and consumers with the intent to provide a certain aggregated load schedule for active power has some objective similarities to controlling a virtual power plant. Within the smart grid domain the volatile character of such a group of distributed energy resources has additionally to be taken into account. On an abstract level, approaches for controlling groups of distributed devices can be roughly divided into centralized and distributed scheduling algorithms.

Centralized approaches have long time dominated the discussion [9], not least because a generator may achieve slightly greater benefit if optimization is done from a global, omniscient perspective [10]. Centralized methods are discussed in the context of static pools of DER with drawbacks and restrictions regarding scalability and particularly flexibility.

Recently, distributed approaches gained more and more importance. Different works proposed hierarchical and decentralized architectures based on multi-agent systems and market based computing [11, 12]. Newer approaches try to establish self-organization between actors within the grid [13–15].

In contrast, VPP are usually operated by a single authority that at the same time is the owner of (and responsible for) all distributed energy resources in this rather static unit ensemble. Here it is quite easy to establish a centralized control instance and moreover: no need arises for a distribution of any generated surplus. In case of a coalition of temporarily pooled but individually owned and operated units (as expected in smart grid scenarios), on the other hand, the question for a fair distribution of jointly earned profit has not yet sufficiently been answered.

2.2 Coalitional Games and Value Distribution

Grouping agents together so as to form coalitions is a major interaction concept within multi agent systems. The usual intention behind this behaviour is that a group of agents might achieve a goal better than a single agent [16–18]. Often,

a group of agents is indispensable, because a single agent alone might not have the capability to achieve the goal. Usually, it is possible to have different groups consisting of a different mixture of agents.

Clearly, each agent (or the associated real world unit) possesses certain traits or benefits that contribute to the overall success of the coalition with different amount. We will not consider different bargaining power in this paper. In some use cases, the utility of the members of a coalition may be easily quantified. For example, [19] studies several examples from the transport sector, with numbers of drivers or trucks and costs expressed in dollar. In the use case studied in this paper, the utility has to be expressed as the contribution that eases a joint planning problem (jointly planning individual load for gaining a wanted aggregated schedule) what has to be expressed in terms of traits and size of individual search spaces of alternative schedules. The actually chosen schedule is not necessarily proportional to the utility of the unit because the richness of offered opportunities (moreover in case of having backup capabilities for later re-scheduling) has to be considered; not the taken choice.

Distributing joint cost is a long discussed topic [20]. The division of surplus has use cases and applications in production [20, 21], electricity pricing [22], public goods [23, 24], and other situations modeled by cooperative games.

A survey of coalition games applied to several use cases in the smart grid can be found in [25]; e.g. some utility functions (analytically calculable) are given but no profit distribution is discussed.

An early application to power planning has been studied in [26]. They used a bilateral Shapley value approach in a multi-agent system for coalition forming when determining cost distribution among beneficiaries of transmission system expansions. In [27] a payment mechanism for VPP is proposed based on the accuracy of individually forecasted power supply. Individual utility of the members to the whole coalition is not considered due to the intractability of calculation.

Intractability in general limits applications of exact Shapley values based distribution schemes. Possible approximation schemes are discussed in the next section.

3 Background

Cooperative game theory is concerned with problems from coalition formation in multi agent systems and desirable properties of such coalitions [3, 28]. Among others, cooperative game theory offers concepts for fair distribution of jointly earned gains among all members of a coalition. The Shapley value [29] is such a concept for fair payoff distribution [19].

If cooperative behaviour is enforced by offering better payoff for an agent within coalitions (groups of players), the competition is between coalitions rather than between individual agents. Such game is said to be a coalition game. If the utility may be exchanged between players (for example by side payments) the game is said to have a transferable utility.

Let N be a set of n players (agents). A coalition game (N, v) with transferable utility for the player set N is characterised by a function v that measures the

worth $v(S)$ of any non-empty subset $S \subseteq N$. In this sense, $v(S)$ denotes the total jointly earned payoff or benefit of a coalition S , that can be distributed among all members of S . Of course, each member makes an individual contribution that has to be considered when determining individual payoff.

The characteristic function v allows to assess the worth of a coalition in the context of a given scenario but gives no hints on how to share it. Shapley [29] defined a value for games that exhibits certain fairness properties [30]. The Shapley value φ_i that is assigned to player i according to a given characteristic function v in a coalition game (N, v) that determines the gain is defined as

$$\varphi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} (v(S \cup \{i\}) - v(S)). \quad (1)$$

The term $v(S \cup \{i\}) - v(S)$ in (1) refers to the marginal contribution of player i to the value of the whole coalition [31].

The marginal contribution defines how much a player i contributes to the overall gain or phrased differently: how important player i is for the coalition. The value assigns a unique distribution of the total generated surplus among all members of a coalition based on the marginal contributions. Shapley values determine a fair distribution.

Especially, Shapley values are the only value distribution scheme that has all of the following four properties (cf. e.g. [19]):

1. **Pareto-efficiency:** The total value of a coalition is distributed among the members: $\sum_{i \in N} \varphi_i(v) = v(N)$
2. **Symmetry:** The value can be determined regardless of the name of the players. If for any two players i and j the following relation holds: $v(S \cup \{i\}) = v(S \cup \{j\})$ for every subset $S \subseteq N$ with $S \cap \{i, j\} = \emptyset$, then $\varphi_i(v) = \varphi_j(v)$.
3. **Additivity:** This property requires for any two games $\varphi_i(N, v)$, $\varphi_i(N, v^*)$ that the following relation holds: $\varphi_i(N, v) + \varphi_i(N, v^*) = \varphi_i(N, v + v^*) \forall i \in N$.
4. **Zero player:** If the marginal value of a player to any possible coalition is zero, this player gains a value of zero: $v(S \cup i) = v(S) \forall S \Rightarrow \varphi_i(v) = 0$.

Calculating Shapley values in general is intractable [32, 33] because the calculation is often #p-complete [34]. The complexity depends on the value function v . There are some special cases where computational feasible calculations or at least approximations of Shapley values are known, e.g. supermodular games [31].

Weighted voting games are an example with computationally hard Shapley values [3]. In such games a coalition gets a reward of 1 if the sum of individual weight values assigned to each agent is larger or equal to some given threshold and 0 otherwise. For a smart grid use case this could translate as follows: if a coalition has to jointly archive at least power q , each generator (or its controlling agent respectively) is assigned a weight of its generation power p_i . A generator coalition is rewarded if $\sum_i p_i \geq q$. An extension to this could demand that the deviation of the joint power to the demanded power is below some given threshold. Another use case would devalue larger deviations from the target

schedule and thus continuously evaluate gains by the actually achieved similarity of demanded and achieved schedule. We will stick mainly with this use case.

Several methods have been proposed to approximate Shapley values for coalition games. Mann and Shapley started with proposing a method based on Monte Carlo simulations in [35], but without suggesting how to draw necessary samples of coalitions; a more comprehensive analysis of sampling approaches can be found in [36]. In [37] a multi-linear extension approach with linear time complexity has been proposed, which has but to be weakened when improving approximation error [38]; [39] proposed a method based on choosing random permutations from coalitions. We will here adapt an randomized approach from [3] with linear complexity.

4 Shapley-Values in Distributed Power Planning Scenarios

4.1 The Planning Scenario

We consider the following power planning scenario within a smart grid: A coalition of (rather small) energy resources is supposed to operate in a way that they jointly produce a wanted active power schedule for a given time frame. The objective is that power production meets power consumption as close as possible at every moment in time. Usually, time is discretized and the energy in every discrete time period (often 15 minutes or shorter) is considered. To this end, we have a demanded load schedule (active power) for a given future time horizon

$$P = (p_1, p_2, \dots, p_n) \in \mathbb{R}^d,$$

with p_i denoting mean electric active power (energy would also do) during the i -th time period for d time periods. This is the active power product that the given group of energy resources is going to deliver according to some market contract. Let

$$s(i) = (p_{i,1}, p_{i,2} \dots, p_{i,n}) \in \mathcal{F}_{(O_0,i)} \subset \mathbb{R}^d$$

denote a schedule of agent i taken from the individual feasible region $\mathcal{F}_{(O_0,i)}$ (containing all operable schedules) resulting from the starting operational state O_0 of the unit of agent i . We are now interested in a set of schedules $\{s(i)|i \in S\}$ that minimizes

$$\|P - \sum_{i \in S} s(i)\| \rightarrow \min \tag{2}$$

with $\|\cdot\|$ denoting some norm that measures the difference between two schedules. We used the Euclidean norm. Phrased informally, each unit from coalition S must choose exactly one schedule from its feasible region of realizable schedules such that all individual schedules jointly resemble product schedule P as close as possible.

The individual contribution to the quality of the solution that an agent offers to the coalition is the individual search space and its abilities, not the finally

picked schedule. This search space determines the set (size, variability and diversity) of alternative choices from that an planning algorithm might choose. In case of a many-objective version of the above sketched problem, cost indicators for a cost effective choice of alternatives would also be part of the search space.

4.2 Distributed Estimation of Shapley-Values

As has been mentioned in the background section, computing Shapley values exactly is computational hard and therefore intractable for larger coalitions of DER. In order to make nevertheless use of these indicators as a base for sharing the surplus, we used a Monte Carlo based estimation scheme after [3].

The method basically works as follows: instead of calculating all marginal contributions to all possible sub-coalitions of a given player j , draw a random sample of n coalitions with size $1, 2, \dots, n$. The average of the marginal contributions of Player j in the random sample is taken as an approximation $\hat{\varphi}_j(v)$ for the Shapley value $\varphi_j(v)$. Obviously, this approximation can be achieved with linear time complexity $\mathcal{O}(n)$.

Next, we have to define the marginal contribution of a DER for a coalition within the sketched smart grid load balancing scenario. Such a coalition is first and foremost aiming at resembling a given load schedule as close as possible. For this reason, it seems appropriate to choose the metric that measures the distance between achieved load and wanted load a basis for our considerations, as this metric is also used as (at least one) objective during optimization. In this way, the marginal contribution is proportional to the difference between the minimal distance that a coalition can achieve with agent j and the minimal distance achieved without agent j :

$$c(j, S) = \frac{1}{\delta_{min}(P, s(j) + \sum_{i \in S} s(i))} - \frac{1}{\delta_{min}(P, \sum_{i \in S} s(i))} \quad (3)$$

with $c(j, S)$ denoting the marginal contribution of agent j to coalition S and $\delta_{min}(\cdot, \cdot)$ denoting the minimal distance (best product resemblance) that a coalition may achieve. The reciprocal of the distance is used, because achieving a small distance is, of course, of a bigger value for the coalition. The schedules $s(i)$ of agents i are chosen in a way that the coalition as a whole gets off best, e.g. in a way that the sum resembles the target load P as close as possible and are therefore the result of an optimization process.

Another possible interpretation would be to define an ϵ as a threshold for allowed average load deviation per time period and assign to an agent a marginal value of 1 if he is a swing player (if the player enables a coalition to fall below the threshold $P - \epsilon \leq \sum_i s(i) \leq P + \epsilon$ when joining the coalition) and 0 otherwise. In this case, the marginal contribution $E\Delta_j^X$ to a coalition of size X can be estimated by [3]:

$$E\Delta_j^X = \frac{1}{\sqrt{2\pi\nu/X}} \int_{(q-w_i)/X}^{(q-\epsilon)/X} e^{-X \frac{(x-\mu)^2}{2\nu}} dx, \quad (4)$$

with the so called quota q that translates to the target load (share of active power of a unit) in our use case, a weight w (here: the power contributions of the units) for each player i and μ and ν as mean and variance of the weights. The Shapley value may then be approximated by [3]:

$$\hat{\varphi}_j(v) = \frac{1}{n} \sum_{X=0}^{n-1} E \Delta_j^X \quad (5)$$

In any smart grid scenario with coalitions of agents trying to assemble a profitable product by scheduling their individual loads, the applicability of Eq. (5) depends on whether the agents are swing players (players enabling a gain for a coalition) or not: depending on market and payoff strategy. In any other case, the Monte Carlo approach that averages the marginal contribution (e. g. Eq. 3) from a sample of coalitions can still be used:

$$\hat{\varphi}_j(c) = \frac{1}{n} \sum_{X=0}^{n-1} c(j, S^X), \quad (6)$$

with S^X denoting a random coalition with size X . The latter case is only applicable if power is planned for a single time period and thus for 1-dimensional schedules. We will now focus on the case with longer planning periods.

In any case, the optimization problem Eq. (2) has to be solved multiple times during the estimation process for calculating marginal contributions.

The task of the optimization process is to determine how good (close in the sense of distance between schedules) a given coalition can adapt the joint schedule to the product. Then, we can compare how good a coalition performs with and without a prospective member and thus how much the agent in question may put forward the group.

In the case considered here, a constraint problem has to be solved. Each energy resource has to restrict its possible operations due to a set of individual constraints. These can be distinguished into hard constraints (usually technically rooted, e.g. minimum and/or maximum power input or output) and soft constraints (often economically or ecologically rooted, e.g. personal preferences like noise pollution in the evening). When determining an optimal partition of the schedule, exactly one alternative schedule is taken from each energy units search space of individually operable schedules (individual feasible region) in order to assemble the desired aggregate load schedule.

In general, any optimization procedure that is able to handle these individual constraints during the search process would do. We used a distributed optimization approach that has been introduced in [40]. In this approach a decoder is harnessed to handle the individual constraints of the energy units by transforming the problem into an unconstrained one. Such a decoder is a constraint-handling technique [41] that maps the constrained problem space to some other not-restricted space where the search operates. In this way, it becomes possible to get every agent a model of the search spaces of all other agents and a decoder that allows for an unconstrained exploration of these search spaces.

To do this, in a first step a surrogate model based on a support vector approach [42] is learned that substitutes for the individually implemented simulation models (of different energy resources) and enables uniform access to the information on which schedules can be realized and which not. In our scenario, we assume an agent in charge of controlling a single energy resource. With such an approach it becomes possible for each agent to solve an optimization problem that involves a parallel search in the search spaces of all other agents in a coalition. Each agent has a model of his own and one for the search space of each other agent and may harness the associated set of decoders to explore these search spaces without having to know the individual constraints. In a round-robin approach each agent tries to improve the overall solution (available for all agents) by changing the own schedule to the one that most improves the solution (after making it feasible with the help of the decoder). A major advantage of this approach is that it can be also calculated by a single agent alone (as central, population based algorithm) as soon as he knows the decoders from all other agents in order to convert solution into feasible solutions. In this way, each agent can do the calculation for optimization during calculating the own Shapley value without having to know anything about the search space structure (or its calculation) of the others.

In this way, each agent can calculate his own Shapley value according to the following process (for the case of Eq. (3)): After choosing a random set of coalitions for the Monte Carlo approach mentioned above, an agent may calculate his marginal contributions by solving the respective optimization problems with and without his own contribution. With the help of the above mentioned optimization approach all calculations that are necessary for calculating an agents own Shapley value can be done by the agent himself, because each agent possesses all search space models from the other agents in the coalition.

Thus, all calculations for Shapley value estimation can be done in parallel, if each agent calculates the own value with the help of the decoders of all others. The amount of necessary communication in distributed environments as well as local computationally complexity depends on the used heuristic for the optimization steps for determining the best achievable aggregated schedule of a coalition to resemble the wanted target schedule.

4.3 Simulation Results

So far, we have tested the approach with simulated μ -CHP (combined heat and power generator); each controlled by an agent.

Of course, each generator has to obey individual constraints such as time varying thermal buffer charging, power ranges, minimum ON/ OFF times, etc. For this reason, we simulated individual plants. For our simulations, we used simulation models of modulating CHP-plants (capable of varying the power level). In order to gain enough degrees of freedom for varying active power, each CHP is equipped with an 800ℓ thermal buffer store. Thermal energy consumption is modeled and simulated by a model of a detached house with its several heat losses (heater is

Table 1. Proportions of profit share approximated with the proposed approach for different scenarios: all equal column 1 and 2; two chp with extra heat demand: column 3; four with extra demand: column 4

CHP No.	CALCULATED	ESTIMATED	2 SPECIAL	4 SPECIAL
1	12.498±4.346E-3	12.497±4.990E-3	14.115±1.123E-1	13.156±2.024E-1
2	12.500±6.735E-4	12.500±3.848E-5	14.270±1.229E-1	13.014±2.341E-1
3	12.503±5.809E-3	12.504±6.326E-3	11.856±4.490E-2	13.301±2.517E-1
4	12.501±1.912E-3	12.502±2.597E-3	11.852±4.504E-2	13.056±2.284E-1
5	12.499±2.023E-3	12.499±1.811E-3	11.960±3.897E-2	11.586±3.772E-1
6	12.500±4.805E-3	12.503±5.826E-3	11.834±4.698E-2	11.916±2.637E-1
7	12.497±4.987E-3	12.497±5.608E-3	12.246±1.749E-2	11.953±2.331E-1
8	12.499±1.845E-3	12.499±2.378E-3	11.868±4.403E-2	12.017±3.061E-1

supposed to keep the indoor temperature on a constant level) and randomized warm water drawing for gaining more diversity among the devices.

For each simulated household, we implemented an agent capable of simulating the CHP (and surroundings and auxiliary devices) on a meso-scale level with energy flows among different model parts but no technical details. All simulations have so far been done with a time resolution of 15 minutes for different forecast horizons. Although, our method is indifferent about any such time constraints. We have run several test series with each CHP randomly initialized with different buffer charging levels, temperatures and water drawing profiles. First, we tested a scenario with identical CHP trying to jointly resemble a given active power schedule. As we used identically parameterized CHP the expectation is that each unit gets the same share of profit (regardless of the actual load distribution). Table 1 shows the result for several runs with 8 CHP, so each of them is expected to get 12.5% of the coalition’s payoff. The distribution key has been calculated by expressing the Shapley value of each CHP as a percentage of the sum of all Shapley values. This approach with an allocation formula is necessary, because in our scenarios Shapley values represent an abstract worth (e. g. min. achievable distance) and do not directly represent a distributable value like money. In this way, we only may indicate a distribution scheme for sharing the profit that is calculated by market, cost, expenses, and prices after delivery.

Table 1 shows the distribution for 8 CHP with calculated (eq. 1) and with linearly estimated (eq. 6) Shapley values. Note that the calculated values are also not precise because they rely on heuristic optimization results, too. This two aspects of estimation have to be distinguished. The linear approximation performs competitively good compared with the exact (modulo the used optimization heuristic) calculation of the values (eq. 1), which would not be tractable for coalitions with sizes beyond about 20 agents. We gained similar results for the use case related to the weighted voting game eq. (5), but with a larger variance. Obviously, this case is much more sensitive to the heuristic that is based on a randomly chosen start configuration of the units in our simulations. In a second scenario (table 1, columns 2,4 special), two and respectively four CHP are given an extra thermal demand during the first hour and a target schedule was used that demands for more power during this period.

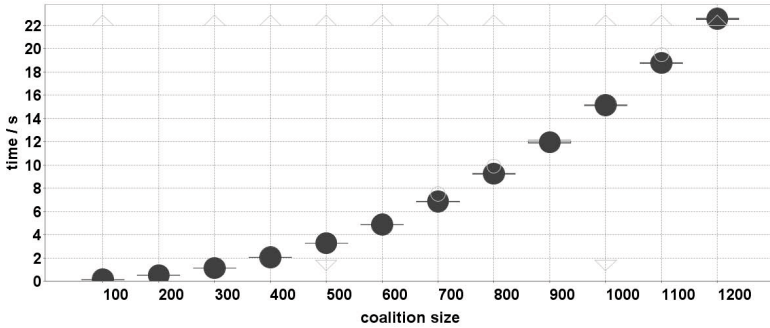


Fig. 1. Computation time for the Shapley value of an agent for different coalitions sizes (Java on a 3 GHz Windows system)

Finally, we scrutinized the computation time for larger coalition sizes. Figure 1 shows the results for coalitions up to 1200 members. As we assume that each agent calculates the own value and the calculations of all agents can be done in parallel, the denoted times are the totally necessary times. As has been discussed earlier, the number of operations for determining the values grows linearly with coalition size if the proposed approximation is applied. One of these operations is: solving the optimization problems that determines the smallest achievable distance to the target schedule for a given coalition. In our case, the complexity for optimization also grows linearly with coalition size. This is due to the used heuristic [40]. Other algorithms might show a different behaviour. Thus, the total complexity for approximating a Shapley value in our case is in $\mathcal{O}(n^2)$ for each single agent. As figure 1 shows, the time consumption is still reasonable considering day-ahead planning and real-time conditions of several hours for the whole negotiation and active power planning process.

In the swing player case eq. (5), complexity would still be in $\mathcal{O}(n^2)$ although all calculations can only be done for one time period at a time and have thus to be repeated for each period.

5 Conclusion and Further Work

Having self-organized coalitions of agents from different owners that plan, produce and sell active power products jointly generated by units from different owners, demands for a scheme for fair distribution of jointly gained payoffs. Shapley values are a well known means of calculating fair shares of payoff for all members of a coalition.

Shapley values are an important conceptual basis for gaining unique and fair distribution keys for profit sharing. Each agent is assigned a share of surplus according to individual marginal contribution. The marginal contributions may be calculated according to an evaluation function that takes into account optimization processes and results in order to determine the capability of fulfilling a active power provision product. In this way, the added value of each agent to

a coalition according to a given power product is expressed as the added value for improving the expected joint optimization result resulting from introducing new degrees of freedom for planning by bringing in individual search spaces.

In this way, each agent is evaluated according to individual capabilities for easing cost effective overall adaption of the group and not for delivering an assigned schedule that (at least in case of using heuristics) partly relies on random and arbitrariness of the scheduling algorithm.

Clearly, the distribution key is only an abstraction that gives hints on how to distribute the surplus of the coalition. Calculating the money value of each agent (or rather of the energy unit the agent is in charge of) has to take into account further considerations of expenses, the course of re-scheduling actions during product delivery, market, and so on.

For now, we have shown that Shapley values may be efficiently estimated with a linear Monte Carlo approximation scheme combined with a heuristic for the worth of a coalition. All calculations may be done fully parallel if all members are trustworthy. In this way, we now have an appropriate tool for a distribution scheme that possesses certain fairness properties from game theory as a essential basis for more sophisticated distribution keys in future developments. Concrete application schemes for such distribution keys are still to be developed.

References

1. Nieße, A., Lehnhoff, S., Tröschel, M., Uslar, M., Wissing, C., Appelrath, H.J., Sonnenschein, M.: Market-based self-organized provision of active power and ancillary services. *IEEE* (June 2012)
2. Bremer, J., Rapp, B., Sonnenschein, M.: Support vector based encoding of distributed energy resources' feasible load spaces. In: *IEEE PES Conference on Innovative Smart Grid Technologies Europe, Chalmers Lindholmen, Gothenburg, Sweden* (2010)
3. Fatima, S.S., Wooldridge, M., Jennings, N.R.: A linear approximation method for the shapley value. *Artif. Intell.* 172(14), 1673–1699 (2008)
4. Mao, Y., Li, M.: Optimal reactive power planning based on simulated annealing particle swarm algorithm considering static voltage stability. In: *Proceedings of the 2008 International Conference on Intelligent Computation Technology and Automation, ICICTA 2008*, vol. 1, pp. 106–110. *IEEE Computer Society, Washington, DC* (2008)
5. Xiong, W., Li, M.: An improved particle swarm optimization algorithm for unit commitment. In: *Proceedings of the 2008 International Conference on Intelligent Computation Technology and Automation, ICICTA 2008*, vol. 1, pp. 21–25. *IEEE Computer Society, Washington, DC* (2008)
6. Pereira, J., Viana, A., Lucus, B., Matos, M.: A meta-heuristic approach to the unit commitment problem under network constraints. *International Journal of Energy Sector Management* 2(3), 449–467 (2008)
7. Guan, X., Zhai, Q., Papalexopoulos, A.: Optimization based methods for unit commitment: Lagrangian relaxation versus general mixed integer programming, vol. 2, p. 1100 (2003)

8. Franch, T., Scheidt, M., Stock, G.: Current and future challenges for production planning systems. In: Kallrath, J., Pardalos, P.M., Rebennack, S., Scheidt, M., Pardalos, P.M. (eds.) *Optimization in the Energy Industry*. Energy Systems, pp. 5–17. Springer, Heidelberg (2009)
9. Tröschel, M., Appelrath, H.-J.: Towards reactive scheduling for large-scale virtual power plants. In: Braubach, L., van der Hoek, W., Petta, P., Pokahr, A. (eds.) *MATES 2009*. LNCS, vol. 5774, pp. 141–152. Springer, Heidelberg (2009)
10. Gatterbauer, W.: Economic efficiency of decentralized unit commitment from a generator's perspective. In: Ilic, M. (ed.) *Engineering Electricity Services of the Future*. Springer (2010)
11. Kamphuis, R., Warmer, C., Hommelberg, M., Kok, K.: Massive coordination of dispersed generation using powermatcher based software agents (May 2007)
12. Kok, K., Derzsi, Z., Gordijn, J., Hommelberg, M., Warmer, C., Kamphuis, R., Akkermans, H.: Agent-based electricity balancing with distributed energy resources, a multiperspective case study. In: *Hawaii International Conference on System Sciences*, p. 173 (2008)
13. Kamper, A., Eßer, A.: Strategies for decentralised balancing power. In: Lewis, A., Mostaghim, S., Randall, M. (eds.) *Biologically-Inspired Optimisation Methods*. SCI, vol. 210, pp. 261–289. Springer, Heidelberg (2009)
14. Mihailescu, R.C., Vasirani, M., Ossowski, S.: Dynamic coalition adaptation for efficient agent-based virtual power plants. In: Klügl, F., Ossowski, S. (eds.) *MATES 2011*. LNCS, vol. 6973, pp. 101–112. Springer, Heidelberg (2011)
15. Ramchurn, S.D., Vytelingum, P., Rogers, A., Jennings, N.R.: Agent-based control for decentralised demand side management in the smart grid. In: Sonenberg, L., Stone, P., Tumer, K., Yolum, P. (eds.) *AAMAS*, pp. 5–12. IFAAMAS (2011)
16. Mas-Colell, A., Whinston, M.D., Green, J.R.: *Microeconomic Theory*. Oxford University Press (June 1995)
17. Kahan, J., Rapoport, A.: *Theories of coalition formation*. Basic Studies in Human Behavior. L. Erlbaum Associates (1984)
18. Osborne, M.J., Rubinstein, A.: *A Course in Game Theory*. 1st edn. The MIT Press (July 1994)
19. Hsu, M.C., Soo, V.W.: Fairness in cooperating multi-agent systems – using profit sharing as an example. In: Lukose, D., Shi, Z. (eds.) *PRIMA 2005*. LNCS, vol. 4078, pp. 153–162. Springer, Heidelberg (2009)
20. Friedman, E., Moulin, H.: Three methods to share joint costs or surplus. *Journal of Economic Theory* 87(2), 275–312 (1999)
21. Sen, A.K.: Labour allocation in a cooperative enterprise. *The Review of Economic Studies* 33(4), 361–371 (1966)
22. Lima, J., Pereira, M., Pereira, J.: An integrated framework for cost allocation in a multi-owned transmission-system. *IEEE Transactions on Power Systems* 10(2), 971–977 (1995)
23. Mas-Colell, A.: Remarks on the game-theoretic analysis of a simple distribution of surplus problem. *International Journal of Game Theory* 9, 125–140 (1980)
24. Champsaur, P.: How to share the cost of a public good? *International Journal of Game Theory* 4, 113–129 (1975)
25. Saad, W., Han, Z., Poor, H.V., Basar, T.: Game theoretic methods for the smart grid. *CoRR abs/1202.0452* (2012)
26. Yen, J., Yan, Y.H., Wang, B.J., Sin, P.K.H., Wu, F.F.: Multi-Agent Coalition Formation in Power Transmission Planning. In: *Hawaii International Conference on System Sciences*, pp. 433–443 (1998)

27. Chalkiadakis, G., Robu, V., Kota, R., Rogers, A., Jennings, N.: Cooperatives of distributed energy resources for efficient virtual power plants. In: The Tenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), May 2011, pp. 787–794 (2011) (event dates: May 2-6, 2011)
28. Rapoport, A.: *N-person game theory. Concepts and applications.* Ann Arbor science library. Univ. of Michigan Pr. (1970)
29. Shapley, L.S.: A value for n -person games. *Contributions to the theory of games 2*, 307–317 (1953)
30. Ma, R., Chiu, D., Lui, J., Misra, V., Rubenstein, D.: Internet economics: the use of shapley value for isp settlement. In: Proceedings of the 2007 ACM CoNEXT Conference, CoNEXT 2007, pp. 6:1–6:12. ACM, New York (2007)
31. Liben-Nowell, D., Sharp, A., Wexler, T., Woods, K.: Computing shapley value in supermodular coalitional games. In: Gudmundsson, J., Mestre, J., Viglas, T. (eds.) COCOON 2012. LNCS, vol. 7434, pp. 568–579. Springer, Heidelberg (2012)
32. Deng, X., Papadimitriou, C.H.: On the complexity of cooperative solution concepts. *Math. Oper. Res.* 19(2), 257–266 (1994)
33. Matsui, Y., Matsui, T.: NP-completeness for calculating power indices of weighted majority games. *Theor. Comput. Sci.* 263(1-2), 306–310 (2001)
34. Valiant, L.G.: The complexity of computing the permanent. *Theor. Comput. Sci.* 8, 189–201 (1979)
35. Mann, I., Shapley, L.: Values of large games, iv: Evaluating the electoral college by monte-carlo techniques. Technical report, Santa Monica, CA: RAND Corporation (1960)
36. Bachrach, Y., Markakis, E., Procaccia, A.D., Rosenschein, J.S., Saberi, A.: Approximating power indices. In: Padgham, L., Parkes, D.C., Müller, J.P., Parsons, S., eds.: AAMAS (2), IFAAMAS, 943–950 (2008)
37. Owen, G.: Multilinear extension of games. *Management Science* 18(5–Part–2), 64–79 (1972)
38. Leech, D.: Computing power indices for large voting games. *Management Science* 49(6), 831–837 (2003)
39. Zlotkin, G., Rosenschein, J.S.: Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domains. In: Proceedings of the Eleventh National Conference on Artificial Intelligence, pp. 32–437. AAAI Press (1994)
40. Bremer, J., Sonnenschein, M.: A distributed greedy algorithm for constraint-based scheduling of energy resources. In: Ganzha, M., Maciaszek, L.A., Paprzycki, M. (eds.) FedCSIS, pp. 1285–1292 (2012)
41. Kramer, O.: A review of constraint-handling techniques for evolution strategies. *Appl. Comp. Intell. Soft. Comput.* 2010, 1–19 (2010)
42. Bremer, J., Rapp, B., Sonnenschein, M.: Encoding distributed search spaces for virtual power plants. In: IEEE Symposium Series in Computational Intelligence 2011 (SSCI 2011), Paris, France (April 2011)

A Trust-Based Approach for Detecting Compromised Nodes in SCADA Systems

Francesco Buccafurri, Antonello Comi, Gianluca Lax, and Domenico Rosaci

University of Reggio Calabria, DIIES Department,
via Graziella, Feo di Vito, 89122 Reggio Calabria, Italy
{bucca, antonello.comi, lax, domenico.rosaci}@unirc.it

Abstract. Nowadays, many critical infrastructures are monitored by SCADA systems processing data obtained by underlying sensor networks. Modern SCADA systems are usually networked, also using wireless connections. Thus, security concerns are crucial when developing SCADA applications, as they are increasingly vulnerable to cyber attacks. In this context, the detection of misbehaving nodes is a key issue, which is in general not easy to address due to the logical and physical high distribution of nodes as well as their complex functions in the network. To deal with the above problem, approaches based on information sharing among collaborative components seem suitable. However, all the past proposals based on information sharing only focus on detecting misbehaving sensor nodes without considering all the other SCADA nodes at any level of complexity. In this paper, we present a trust-based approach to detecting high-level compromised nodes in a SCADA system that is based on a competition among agents associated to nodes. Some preliminary experiments we have performed show promising results of the proposed approach in terms of effectiveness and efficiency.

1 Introduction

Nowadays, most manufacturing or processing plants are supported by a Supervisory Control And Data Acquisition (SCADA) system, which is a computer-based control system allowing system operators to oversee a variety of processes. By using distributed electronic controls and sensors to perform batch or repetitive tasks, SCADA alerts the operator if some system component needs attention or has exceeded pre-set parameters. While the first SCADA systems held all operations in one computer (generally a mainframe) and SCADA functions were limited to only monitoring sensors, the later SCADA systems use a distributed architecture since they often share control functions across multiple smaller computers. Moreover, if in the first distributed SCADA systems the nodes were connected by Local Area Networks, current SCADA systems are usually networked, communicating through Wide Area Networks, and often the clients can access the system using Internet and the Web [9], also via a wireless connection [7]. As an example of a modern Web-based SCADA system, in Figure 1 is represented the architecture of the Broadwin WebAccess networking [2].

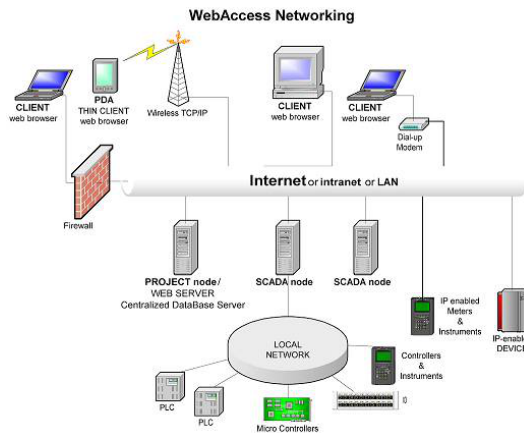


Fig. 1. Broadwin WebAccess: An example of Networked SCADA architecture

Currently, SCADA systems are built upon a Wireless Sensor Network [12]. In such a context, security concerns are key issues when developing SCADA applications, as they are vulnerable to cyber attacks. In many current SCADA systems, the use of a Virtual Private Network (VPN) is the only security protection adopted, but the possibility of physical access to SCADA-related network jacks and switches provides the ability to totally bypass firewall and VPN security. This problem is best addressed by endpoint-to-endpoint authentication and authorization provided by cryptographic techniques, but encryption is not suitable for sensor devices because resource restrictions would prevent strong encryption schemes, and low-cost side-channel attacks are sufficient to reveal the private keys [13]. Other techniques for implementing security in SCADA systems, presented to the general security community, address the problem only focusing on the sensor network underlying the global system, proposing secure routing protocols. For example, in [5], an architecture to stimulate a correct routing behavior is presented, where each node receives a per-hop payment in every packet it forwards. Nodes store this payment information in an internal counter and this information is shared by nodes that directly interact, introducing a cooperation element in the security mechanism. In [10], the authors present an approach for mitigating routing misbehavior by detecting non-forwarding nodes and rating every path so those nodes are avoided when the routes are recalculated. In this approach, the non-routing nodes are not included in routing paths, since they are not going to cooperate, but they still can ask others to forward their messages. This scheme detects the misbehavior but it does not isolate it. Also the approach proposed in [4], introduces a secure routing protocol, called CONFIDANT. CONFIDANT operates over the chosen routing protocol and makes misbehaviour less attractive for the nodes than proper routing. In particular, the nodes watch their neighbors for bad behaviour and take this behavior into account by a local reputation system.

Moreover, the nodes can also inform their trusted neighbors on misbehaving nodes. Another similar approach, that uses a reputation system, is that presented in [11]. However, differently from CONFIDANT, in this case the reputation system is not local, but global: some *reputation* information about the sensor nodes is transmitted all over the network, in order to warn all nodes about misbehaving nodes. In this approach, the compromised nodes are detected and isolated, since their reputation is rated as low.

1.1 Contribution

In any case, all the above approaches work at the level of sensor networks, without involving the high-level SCADA nodes. In this paper, we argue that designing a detection strategy that takes all the SCADA nodes (not only the sensors) into account, can introduce two main advantages: (i) it is possible to immediately detect attacks that are directly addressed to high-level SCADA nodes; (ii) it is possible to use different software components for high-level nodes (that are generally fully-equipped PC) and for lower-level nodes (that have limited resource).

Observe that these two levels of components in SCADA systems, namely, (i) the underlying sensor network and (ii) the other high-level of supervision and control, are not equivalent from the viewpoints of security problems and possible countermeasures. For example, a SCADA system may collect a number of measures from different sensors, and may elaborate them, so that a simple detection of outliers in the temporal sequences of sensors (as in [11]) is not immediately extensible to the other high SCADA levels, since the misbehaviour of a SCADA node could be much more complex with respect to that of a sensor node. Therefore, the aforementioned approaches working at sensor network level are not immediately applicable to the whole SCADA system.

On the basis of the above considerations, we propose a technique for detecting compromised nodes in a SCADA system that exploits a trust-based strategy derived from the research field on competitive agents. In our approach, each SCADA node is associated with a software agent, called *control agent*, and control agents are in competition with each other in a game. Each control agent sends control requests to other agents and based on the quality of the responses computes some trust measures that contribute to determine the *global reputation* of each agent. The SCADA nodes associated to the agents having low global reputations are considered as possible compromised nodes. Similarly to other approaches as [5,11], our proposal is based on sharing some individual information (trust measures, in our case) among system nodes, exploiting the distributed architecture. However, differently from the past approaches, we propose to involve in the distributed computation the SCADA high-level nodes (and thus indirectly also the controlled sensor nodes). Therefore, we have the possibility to use a sophisticated trust model, representing both the trust that a node A has in another node M based on its direct past experience (reliability), and the trust about M that A derives from opinions coming from other nodes (reputation). For this purpose, taking inspiration from a previous proposal [14], where a general trust model is introduced in order to study how to combine reliability

and reputation, we introduce a trust-based framework specifically conceived for SCADA systems, using both reliability and reputation, aiming at detecting and isolating compromised SCADA nodes.

Several metrics and techniques to measure reliability and reputations have been proposed in the literature [15,16]. Furthermore, some of these approaches deal with the integration of reliability and reputation into a single synthetic measure [8,17,6,3].

Indeed, it is important to remark that the choice of a correct integration between reliability and reputation is a key issue when using a global trust measure and this choice strictly depends on the characteristics of the application environment. Our model leaves the possibility to design the security mechanism adopting the most suitable combination of these two trust measures, introducing a complete flexibility with respect to the particular SCADA system which the approach is applied to.

Note that our approach could be usefully integrated with any of the past proposals for detecting misbehaving sensor nodes. Indeed, if our approach detects a compromised SCADA node that controls a set of underlying sensor nodes, an approach as those proposed in [4,11] can be used to detect the compromised sensors.

To the best of our knowledge, our proposal is the first attempt to use a trust-based approach for detecting and isolating compromised nodes in a SCADA system, working at high level. As a consequence, our preliminary experimental evaluation described in Section 4 only focuses on studying the performances of our approach, since no immediate comparison with other techniques is possible. However, in our ongoing research, we are studying to design reasonable extensions of the available reputation-based methods for sensor networks to the high level of SCADA nodes, making possible a comparison with our approach.

The paper is organized as follows. In Section 2, we introduce the scenario we deal with as well as a sketch of our proposal. In Section 3, we describe the adopted trust model and our method for detecting compromised nodes. Then, in Section 4, we present the experimental campaign we have performed to validate our proposal, which shows the capability of our approach to effectively and efficiently individuate compromised nodes. Finally, in Section 5, we draw some conclusions and discuss our ongoing research.

2 An Overview of the Proposal

A SCADA system is composed of a set of SCADA nodes where each SCADA node manages a set of sensors. Nodes provide information such as sensor readings. Each requested information belongs to one of the types defined for the application domain (e.g., in a SCADA monitoring a Water Treatment and Distribution System, *flow level*, *flow pressure*, *pump speed*, etc.) If the required information involves measures derived from sensors falling under its competence, a node can directly provide the response. Otherwise, the node has to propagate the request to other nodes in order to provide the response. An example of a

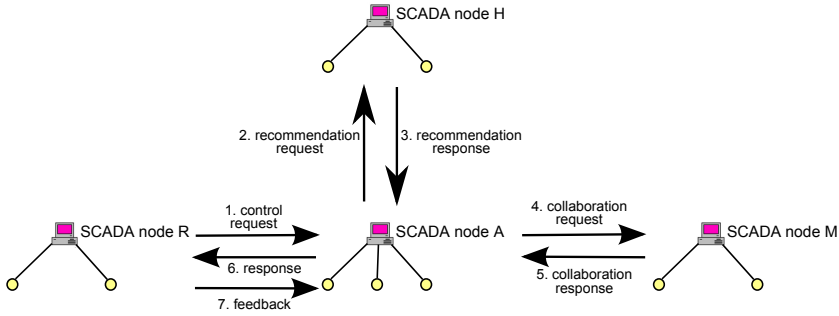


Fig. 2. An example of interactions among agents

SCADA system with four nodes is provided in Figure 2. Here, the SCADA node A manages three sensors, while the other nodes are responsible of two sensors respectively.

In order to detect compromised nodes, we propose an agent-based approach. In our proposal a software agent, called *security agent*, is associated with each SCADA node. Moreover, a central software component, called *security agency*, is used to monitor the behavior of SCADA nodes. These two components, the set of agents and the agency, implement the *security subsystem* which has an important role in our framework. Each agent associated with a SCADA node has a different *expertise* about each information type T , due to its particular capability to provide that information. Indeed, it is possible that the agent can directly obtaining that type of information, so that it behaves as an expert about T , while it is also possible that it has to require the collaboration of other agents. Moreover, the expertise of an agent about an information type can vary in time, due to possible changes in the SCADA system configuration.

In our proposal, we suppose that each agent R periodically sends suitable *control requests* to some other agents. Such requests are used to evaluate the trustworthiness of the other agents. To this end, these requests are related to information already known by the agent. We make an inherent assumption that nodes have a high level of redundancy in sensing coverage, so that other nodes can evaluate the information quality. Assume that the agent A receives a control request. If A is not able to directly provide a response to this request, then it forwards the request to the agents considered the most suitable, based on a trust model. The number of agents to be contacted by A is a design parameter of the security subsystem and can be suitably tuned to avoid an excessive traffic in the communication network. The selection of the most suitable agents is based on the effectiveness shown in the past by the agents in the information type T which the request falls into. During each temporal step, the agents of the security subsystem can interact with each others, in order to exchange information. The interactions among agents follow the protocol described below:

- In order to provide a requesting agent R with a response to a request of a given type T , an agent A may decide to require the collaboration of another

agent M . Before requiring this collaboration, A can ask a third agent H for a *recommendation* about the expected capability of the agent M in providing information of type T . A recommendation is an evaluation of the quality generally associated with the responses provided by M . The recommendations coming from other agents are used by A for updating its internal *reputation model* (see Section 3.1). In other words, A uses the gossips coming from the other agents in order to determine the reputation of M in the community.

- At the end of the step, the agent A receives a feedback from the agent R . The feedback contains an evaluation of the quality of the response and consequently informs A about the quality of the contributions given by the contacted agents to provide the response. This way, A can use the feedback to update its internal *trust model* about the agents of the system.

The overall process that leads to provide the requesting agent with a response is logically decomposed into 7 phases, graphically represented in Figure 2, namely:

1. the requesting agent R submits a control request to the agent A ;
2. A requires recommendations to another agent H about M ;
3. H provides the recommendation of M ;
4. A requires the collaboration of M to obtain the information asked by R ;
5. M provides this information;
6. A provides R with the response to the control request;
7. R provides A with a feedback.

3 Trust-Based Detection of Compromised Nodes

This section describes the trust model and the mechanism that are exploited to detect compromised nodes.

3.1 The Trust Model

The trust model adopted in our approach is strongly based on three trust measures, namely reliability, reputation and preference.

In particular, given two agents A and M , the reliability that A assigns to M represents the subjective measure of the trust that A has in another agent. It comes from the past direct experience of A with M .

Moreover, the reputation that A assigns to M represents a measure of trust that does not depend on past direct experiences. Instead, it is based on some recommendations coming from other agents of the community. Observe that although the reputation is not based on a subjective evaluation of the agent, it is not an objective measure, since the agent A computes the reputation of another agent M independently of how the other agents compute the reputation of M .

Finally, the preference that the agent A assigns to the agent M is obtained by suitably taking into account both the reliability and the reputation perceived by A .

We show how the above trust measures are evaluated by each agent basing on the received recommendations and the strategy defined in our proposal.

At the beginning, since the agent has not had any interaction with the other agents and thus it has no knowledge about the environment, the three trust measures assigned to any other agent can not be evaluated (we can say their value is null).

Now we describe how the trust measures of an agent w.r.t. any other agent are computed and updated.

Consider the i -th step carried out by the agent A (recall the agent interactions described in Section 2). Here, the agent A is required to provide the sensor measures S_1, \dots, S_s of type T , where the type T is related to an information not directly provided by A . As a consequence, A thinks of using the collaboration of another agent M . Before exploiting this collaboration, A requires to other agents a recommendation concerning the expected capability of the agent M in providing information of type T . Such recommendations are of the form $RC = \langle M, T, V \rangle$, where M is the *subject*, T is the information type, and $V \in [0, 1]$ is the *value* of RC . The higher the value of a recommendation $RC = \langle M, T, V \rangle$, the higher the expectation that M is able to respond correctly about information of type T .

After the request is forwarded from A to M and M provides the response, A receives from R a feedback about the quality of the response. Each feedback is a real number belonging to $[0, 1]$. A feedback equal to 0 (1, resp.) means minimum (maximum, resp.) quality of the response. Let FB_1, \dots, FB_s be the feedbacks which evaluate the quality of the responses given by M to the requests S_1, \dots, S_s (recall, of type T).

We show how, at the end of this step, the trust measures are updated.

- **Reliability.** At the i -th step, the reliability $RL^i(A, M, T)$ assigned by A to M in information of type T is computed as:

$$RL^i(A, M, T) = \alpha \cdot RL^{i-1}(A, M, T) + (1 - \alpha) \cdot \frac{\sum_{1 \leq j \leq s} FB_j}{s}$$

where $RL^{i-1}(A, M, T)$ is the previous value of reliability assigned by A to M in information of type T and α is a real value belonging to $[0, 1]$ representing the importance given to the past evaluations of the reliability with respect to the current evaluation (in other words, α measures the importance given to the *memory* with respect to the current time).

Observe that, in case this is the first interaction of A with M , the value of α is set to zero since no past evaluation can be used when updating the reliability.

- **Reputation.** The recommendations on M about information of type T received from the other agents at this and at the previous steps are used by A to evaluate the reputation of the agent M in information of type T .

Denoting by V_1, \dots, V_r the values of such recommendations, the reputation $RP^i(A, M, T)$ assigned by A to M in information of type T at the i -th step is computed as:

$$RP^i(A, M, T) = \frac{\sum_{1 \leq j \leq r} V_j}{r}$$

In words, A computes the reputation of M in information of type T as the mean of all the recommendation values received from the other agents of the community concerning M .

- **Preference.** Finally, at the i -th step, the preference $PR^i(A, M, T)$ assigned by A to M in information of type T is computed as:

$$PR^i(A, M, T) = \beta \cdot RL^i(A, M, T) + (1 - \beta) \cdot RP^i(A, M, T)$$

where β is a real value belonging to $[0, 1]$ which is used to weight the importance of the reliability with respect to the reputation.

The computation of the value of the three trust measures is used for two purposes: the former is to select the most suitable candidates to require a collaboration and this is done by taking into account the value of preference. This issue is not the focus of the paper and will not further discussed. The latter purpose, which is a key issue in this paper, is to detect compromised nodes and will be dealt with in the next section.

3.2 Detection of Compromised Nodes

At the end of each step i , each agent sends to the security agency (see Section 2) the value of the preference assigned to the other agents. The security agency, for each agent and for any information type T , computes a global measure of its reputation in the agent community, called *global reputation* in information type T . Let A_1, \dots, A_p be all the agents that have sent preferences at the step i to the security agency about M in information of type T . The *global reputation* $GR^i(M, T)$ at the i -th step of the agent M in information of type T is defined as:

$$GR^i(M, T) = \frac{\sum_{1 \leq j \leq p} PR^i(A_j, M, T)}{p}$$

i.e., the arithmetic average of all the received preferences.

If the value $GR^i(M, T)$ is lower than a pre-determined threshold τ , the security agency generates an alarm related to the SCADA node associated with the agent M regarding the information type T , which will be subsequently managed by the SCADA system administrators. Note that the value chosen for the threshold τ is crucial in the detection procedure. A too high value for τ could generate some false positive, whereas a too low value for τ could generate some false negative. The choice of τ can be accurately done by the administrator of the SCADA system, based on her experience and on the typology of the SCADA application domain.

4 Evaluation

In this section, we describe some experiments we have performed in order to evaluate both effectiveness and efficiency of our approach in detecting compromised nodes of a SCADA system. The experiments have been carried out by using a prototypical simulator of the scenario described in Section 2. We have built this simulator as an extension of the well-known **Agent Reputation and Trust** (ART) platform [1] that allows the simulation of a competition between agents equipped with a trust model.

Our prototype simulates the behaviour of the SCADA agents of the security subsystem. In particular, each simulated agent has a specific *expertise* which is a real value ranging in $[0, 1]$ assigned by the simulator. For simplicity, in this simulation we have assumed that only one information type exists. The error in responding to a control request generated by the simulator for an agent having an expertise e is described by a normal distribution with mean 0 and standard deviation equal to $1 - e$. In other words, the higher e , the more precise the responses of the simulated agent. The agent's expertise does not change throughout the game and the agents know their levels of expertise. Moreover, the simulator does not inform agents about the other agents' expertise levels. The true values of the control requests presented by the agents to other agents are generated uniformly at random.

We have constructed two types of agents, namely the *normal agent*, representing agents associated with non compromised SCADA nodes, and *compromised agent*, representing agents associated with compromised SCADA nodes. The simulator assigns to a normal agent an expertise generated by a uniform distribution ranging in $[0.8, 1]$, while a compromised agent is provided with an expertise generated by a uniform distribution ranging in $[0, 0.4]$. This way, the normal agent will respond to the control queries with high precision, while the compromised agents will give bad responses. Each agent is equipped with the trust model described in Section 3.1. Both the parameters α and β are set to 0.5, where $\alpha = 0.5$ assigns the same importance to old and new reliability values, while $\beta = 0.5$ assigns the same importance to both reliability and reputation. Moreover, in each simulation step, an agent executes a number n_r of recommendation requests. Each request concerns an opinion about n_t distinct agents. In our experiments, both n_r and n_t have been set to the 5 percent of the agent population, excepting the last experiment where they vary. This relatively small number of allowed requests allows us to avoid an overwhelming amount of control messages in the network, simulating a realistic situation.

In the first experiment, we have run 100-step simulation on a population of 100 SCADA agents composed of 80 normal agents and 20 compromised agents. As a result, we observe that at the end of the simulation all the 20 compromised agents have obtained the 20 worst scores in terms of global reputation. In particular, the compromised agent with the best result has obtained a global reputation equal to 0.35, while the normal agent with the worst result has achieved a global reputation equal to 0.42. The interesting result is that, looking at the ordered score list of the agents, the highest gap between contiguous differences is just the

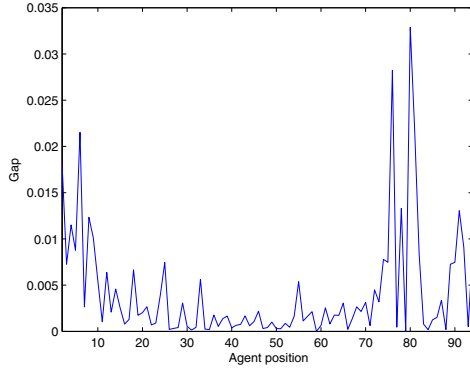


Fig. 3. Contiguous differences in the agent score list

one obtained between the normal agent with the worst result (i.e., 0.42) and the compromised agent with the best result (i.e., 0.35) – see Figure 3. This allowed us to set the value $\tau = 0.40$ as a reasonable threshold to distinguish compromised SCADA nodes from nodes that are not compromised, stating that a SCADA node associated with an agent having a value of global reputation lower than τ will be considered as compromised, while a SCADA node associated with an agent having a value of global reputation greater than or equal to τ will be considered as not compromised. We remark that the issue of correctly tuning the threshold certainly merits a further deepening, which we plan to do in a next phase. Indeed, in this first experimental campaign we focused on the capability of the method of placing the compromised nodes, say k their number, exactly in the bottom- k positions of the score list. In other words, the experimental campaign should be interpreted more as a way to validate an approach instead than a conclusive real-life tool for detecting compromised nodes. Indeed, methods more complex than the simple tuning of a static threshold could be adopted (and we plan to study this issue in our future work) in order to fortify the detection method, like analysis about the dynamics of scores during the steps of the simulation, etc.

In Figure 4, we have plotted the number of false positives (SCADA nodes that are erroneously classified as compromised) and false negatives (SCADA nodes that are erroneously considered as not compromised) for each step of our simulation. We observe that both false positives and false negatives are no longer present after the step 45. In particular, after the first 30 steps, in which the random component of the simulation causes the presence of some (small) errors, the number of false negatives rapidly decreases, while the false positives are already absent after the step 10. Observe that the value of the threshold is in general depending on the percentage of compromised nodes. However, this fact does not inhibit the possibility of detecting compromised nodes, since this can be done by analyzing contiguous differences in the agent list.

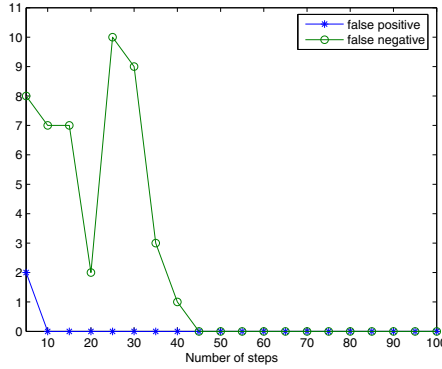


Fig. 4. Number of false positives and false negatives for each step of the simulation

In order to evaluate how the performance of the method depends on the agent population size, we have performed a second experiment, by running a number of further simulations, each corresponding to a population having a different number of agents n_a , namely 60, 80, 100, and 120 agents. We have plotted in Figure 5 the number of steps n_s needed to completely distinguish compromised and not compromised nodes for different values of n_a . We can see that n_s increases with n_a (almost) linearly, showing that the grow of the agent population size is not a critical issue. This intuitively derives from the fact that the number of recommendation and opinion requests of each agent in our simulation linearly increases with the population size. Since this choice seems reasonable to cover the whole agent space in an acceptable number of steps, we can conclude that our approach achieves such a good effectiveness paying an acceptable price in control traffic overhead.

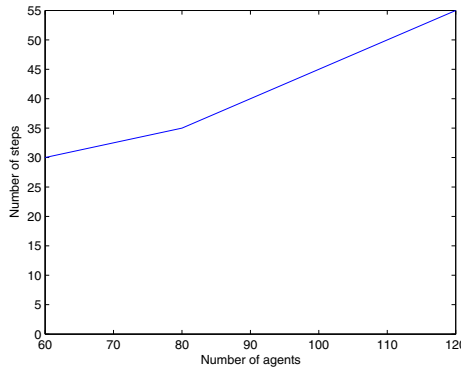


Fig. 5. Number of steps vs different agent populations

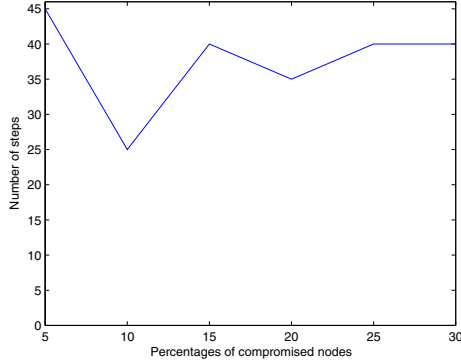


Fig. 6. Number of steps vs percentages of compromised agents

We have performed a third experiment aiming at studying how the efficiency of the system is influenced by the number of compromised nodes. We have performed other 6 simulations on a population having a global number of 100 SCADA nodes, with different percentages of compromised nodes, namely 5, 10, 15, 20, 25 and 30 percent. We have plotted in Figure 6 the number of steps necessary to completely distinguish compromised and not compromised nodes, for different percentages of compromised agents. We observe that the approach is not significantly sensitive to the variation of this percentage, being capable of distinguishing compromised and not compromised nodes with a number of steps comprised between 25 and 45 steps. This good behaviour can be intuitively explained considering that a different number of compromised nodes does not imply a different number of operations for the agents of the control subsystem for exploring the agent space in order to achieve sufficient trust information. In other words, the computation time mainly depends on the size of the agent space (as seen in the second experiment) and only marginally on the number of compromised nodes. As a final experiment, in order to study how the capability of detecting compromised nodes depends on the number of requests sent from an agent, we have performed further simulations (for a population of 100 nodes, with 20 percent of compromised nodes), each corresponding to different values of n_r and n_t . We have set n_r and n_t to the values 2, 5, 7, 15, 25, and 30.

In Figure 7, we have plotted the number of steps necessary to completely distinguish compromised and not compromised nodes, for different percentages n_r of opinion/recommendation requests. The results clearly show that, for $n_r > 5$ an increment of the percentage n_r does not imply significant modifications of the number of steps. This saturation effect can be intuitively explained by considering that once we use an n_r sufficient to explore the whole agent space in a given number of steps, an increment of n_r is not useful to improve the system performances. Then, this result shows that it is necessary to accurately choose the value of n_r in order to avoid a useless waste of resources.

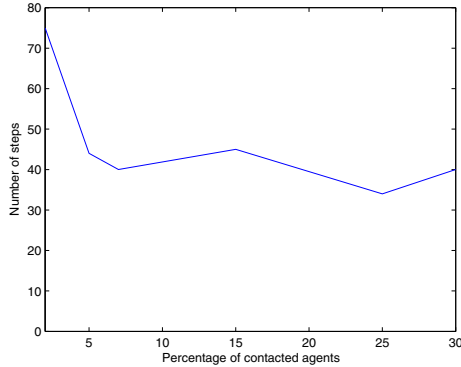


Fig. 7. Number of steps vs n_r and n_t

5 Conclusion

Usually, SCADA sensor networks are composed of sensor nodes with severe resource limitations, joined to the possibility to access physically to the nodes, which makes them highly vulnerable to cyber attacks. In this context, the issue of detecting misbehaving nodes is obviously very important to react in time to an attack. Approaches presented in the literature mainly face this problem by using information sharing at level of sensor network. However, these techniques cannot be immediately extended to high-level SCADA nodes, which perform more complex operations than simply providing data. On the other hand, high-level SCADA nodes do not have strict computational limitations, thus they could be good candidates to implement more sophisticated detection techniques, appropriate for a distributed environment as that of SCADA systems. In this paper, we have presented a trust-based framework for detecting and isolating compromised nodes in the high-level of a SCADA system. Our framework is based on a multi-agent security subsystem that manages a social game involving all the high-level SCADA nodes. The nodes that obtain the worst reputation rates are considered as compromised. The use of a multi-agent system allows us to completely distribute the additional computational effort on the whole SCADA network, making our approach sufficiently efficient and scalable. Moreover, the introduction of the social game makes possible to compute the trust measures by exploiting a sort of gossips among the agents, which avoids an onerous, direct and continuous interrogation of the single nodes for detecting possible misbehaviour. Some preliminary experiments we have performed show a promising effectiveness of the approach. As for our ongoing research, we are planning to validate our approach by applying it to large SCADA networks, simulating a set of realistic common attacks.

Acknowledgements. This work has been partially supported by TENACE PRIN Project (n. 20103P34XC) funded by the Italian Ministry of Education, University and Research.

References

1. ART Testbed (2013), <http://megatron.iiaa.csic.es/art-testbed>
2. Broadwin WebAccess (2013), <http://broadwin.com/features.htm>
3. Buccafurri, F., Comi, A., Lax, G., Rosaci, D.: Enhancing Trust-Based Competitive Multi Agent Systems by Certified Reputation. In: Meersman, R., Panetto, H., Dillon, T., Rinderle-Ma, S., Dadam, P., Zhou, X., Pearson, S., Ferscha, A., Bergamaschi, S., Cruz, I.F. (eds.) OTM 2012, Part II. LNCS, vol. 7566, pp. 855–862. Springer, Heidelberg (2012)
4. Buchegger, S., Boudec, J.L.: Performance analysis of the CONFIDANT protocol. In: Proceedings of the 3rd ACM International Symposium on Mobile Ad hoc Networking and Computing, pp. 226–236. ACM Press, Lausanne (2002)
5. Buttyan, L., Hubaux, J.: Stimulating cooperation in self-organizing mobile ad hoc networks. *Mobile Networks Applications* 8, 579–592 (2003)
6. Garruzzo, S., Rosaci, D.: The Roles of Reliability and Reputation in Competitive Multi Agent Systems. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 326–339. Springer, Heidelberg (2010)
7. Goel, A., Mishra, R.S.: Remote Data Acquisition Using Wireless - SCADA System. *International Journal of Engineering* 3(1), 58–65 (2008)
8. Huynh, T.D., Jennings, N.R., Shadbolt, N.R.: An Integrated Trust and Reputation Model for Open Multi-Agent System. *Aut. Agent and Multi Agent Sys.* 13, 119–154 (2006)
9. Kirubashankar, R., Krishnamurthy, K., Indra, J., Vignesh, B.: Design and Implementation of Web Based Remote Supervisory Control and Information System. *International Journal of Soft Computing and Engineering* 1(4), 43–51 (2011)
10. Marti, S., Giuli, T.J., Lai, K., Baker, M.: Mitigating routing misbehavior in mobile ad hoc networks. In: Proc. of the 6th Annual International Conference on Mobile Computing and Networking, pp. 255–265. ACM Press, Boston (2000)
11. Moya, J.M., Araujo, A., Bankovic, Z., de Goyeneche, J., Vallejo, J.C., Malagon, P., Villanueva, D., Fraga, D., Romero, E., Blesa, J.: Improving Security for SCADA Sensor Networks with Reputation Systems and Self-Organizing Maps. *Sensors* 9, 9380–9397 (2009)
12. Patil, U.S.: Study of Wireless Sensor Network in SCADA System for Power Plant. *International Journal of Smart Sensors and Ad Hoc Networks* 1(2), 41–44 (2011)
13. Ravi, S., Raghunathan, A., Kocher, P., Hattangady, S.: Security in embedded systems: design challenges. *Transactions on Embedded Computer Systems: Design Challenges* 3, 461–491 (2004)
14. Rosaci, D.: Trust Measures for Competitive Agents. *Knowledge and Information Systems* 28(46), 38–46 (2012)
15. Sabater-Mir, J., Paoulucci, M.: On Open Representation and Aggregation of Social Evaluation in Computational Trust and Reputation Models. *International Journal of Approximate Reasoning* 46(3), 458–483 (2007)
16. Sabater, J., Sierra, C.: Review on Computational Trust and Reputation Models. *Artificial Intelligence Review* 24, 33–60 (2005)
17. Xiong, L., Liu, L.: PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. *IEEE TKDE* 16(7), 843–857 (2004)

Unified Energy Agents as a Base for the Systematic Development of Future Energy Grids

Christian Derksen¹, Tobias Linnenberg², Rainer Unland¹, and Alexander Fay²

¹DAWIS, Universität Duisburg-Essen, Schützenbahn 70, 45127 Essen
{Christian.Derksen,Rainer.Unland}@icb.uni-due.de

²IfA, Helmut-Schmidt-Universität Hamburg, Holstenhofweg 85, 22043 Hamburg
{Tobias.Linnenberg,Alexander.Fay}@hsu-hh.de

Abstract. The need for the application of software agents and agent-technologies in highly diversified future energy grids is widely accepted today. Nevertheless, the very general concept of the agent paradigm still leads to misunderstandings and to the fact that agents are meant and utilized for very different tasks. Accordingly, the approaches that were presented in the Smart Grid area have major weaknesses in terms of comparability and a subsequently large-scale use. We claim that the introduction of a unified definition of an Energy Agent will help to create a coherent picture that can accelerate further discussions and the conversion of the energy supply. Considering a development cycle that consists of modeling and implementation, simulation, test-bed application and the deployment to real systems, we present here our definition of an Energy Agent that takes into account the law of conservation of energy. Further, we present a classification of Energy Agents according to their sophistication and integration level and outline the need for individual but standardized energetic option models.

1 Introduction

In the course of the European and global energy-revolution some problems weaved aside as side issues at the beginning now turn out to be the major challenges. One of the most imminent threats is the introduction of a vast amount of isolated and incompatible control applications obstructing a sustainable integration of energy systems into a comprehensive energy control solution.

Due to the highly diversified nature of tomorrows' energy grid, new control solutions have to be found that tackle the rising demand for coordination. Coping with the challenge of fluctuating electricity production by a growing number of ever shrinking energy converters, dependent on meteorological factors as wind or sun, or heat-operated combined heat and power plants, already strains conservative control solutions. Meeting the high requirements with regard to availability of electricity supply, minimization of frequency- and voltage-deviations and other regulatory and socio-economic factors does finally overstrain these control concepts as we may find them in todays' electricity grids. While the physical grid infrastructures and the processes therein become more and more complex, the control solutions are lagging far behind.

The needed change in paradigms has not yet left well controlled laboratory-like setups, where visionary thinking is allowed and contained in restrictive boundaries. In these setups different approaches have been implemented to approach the general problem of an ever growing complexity in number and solutions.

Recent literature presents several Smart Grid concepts. Most of them focus either on stabilizing measures as needed for grid recovery during abnormal load conditions [1], the pooling of different types of energy converters to virtual power plants [2], or the control of connected energy consumers by means of demand side management [3]. All those efforts are united by one drawback - the central control. Today's central control solutions are generally limited in their number of nodes [4]. Due to the recent shift towards an increasing amount of small power plants it seems to be more promising to examine grid control concepts, featuring decentralized decision [5], [6].

One of the key features of Multi Agent Systems (MAS) is their proven reliability in the field of distributed problems. They are typically made up of a high number of basic components which are easily replaceable as each one is a separate entity. This high number of small actors gives a better tolerance towards failures in single modules and will therefore facilitate the compensation of grid failures or undesired grid conditions. In case of abnormal grid conditions it is possible to separate the affected regions from the grid and put them into a so-called 'islanding mode', while stabilizing the detached area allows reconnecting. This implies that the area in question is self-sustainable in regard to its energy needs [7], [8].

Besides all these benefits, there are some downsides, connected with the use of decentralized decision making in the context of energy management: All multi-agent based power control systems in literature known to the authors (e.g. [9], [10], [11]) do apply proprietary data exchange formats and architectures. This leads to compatibility and comparability issues. Even though a multitude of available standards like *IEC61850* or the *Common Interface Modell (CIM)* and proprietary solutions for the different aspects of data exchange and control and command needs in the energy domain are available, none of them is totally suited for the needs of distributed processing and control.

Furthermore do most of these approaches only focus on one form of energy, omitting the potential of solving problems common to other energy domains. As we can find a volatile energy demand and the tracking and quantification of energy flows not only in the domain of electrical energy, it would be desirable to implement one heuristic capable of solving problems common to the management processes of several forms of energy.

Even though the agent based *DEcentralized MARket based POver control System (DEMAPOS)* [12], developed at the Helmut Schmidt University in Hamburg, is capable of tackling some of the above mentioned challenges in regards to electricity networks, several pitfalls and difficulties, common to most alike systems, were discovered in the process of merging it with University Duisburg-Essen's *Agent.GUI* [13] multi-domain simulation environment. The major challenge was found when designing the interfaces between the control and the simulation.

To overcome these problems in the design process and the final implementation we suggest in the following the design and utilization of a generalized Energy Agent based on a unified agent model throughout all phases of the development cycle. This agent features standardized interfaces and inner workings, which will be able solve

the compatibility- and comparability issues not only for the domain of electrical energy, but also for the associated fields of gas-, water- and heat distribution. A differentiation between the agent itself and the surrounding system in the context of a desirable development process will be pointed out. It will be indicated on how to use the same piece of software throughout the entire development cycle - from concept to realization. Further individual problem areas connected with energy distribution and management will be considered as well. The structure of the paper is as follows: in section 3 a short introduction to the theoretical background is given, explaining the agent concept, its communication needs and the thereof derived requirements towards a testing environment. Afterwards the problem statement is specified on the basis of some examples. In section 4 a prototypic design process is introduced and in section 4 the thereof resulting Energy Agent is discussed. Section 5 presents an option model to be used in combination with the Energy Agent. A short synopsis concludes the paper.

2 Theoretical Backgrounds and Problem Statements

For the purpose of this paper an agent can be seen as an autonomous computational entity that is situated in an arbitrary environment and that is capable of autonomous interactions with this environment. Other optional properties and abilities that are also often associated with intelligent agents are their social ability, a reactive or proactive behavior and the capability to learn (e.g. [14]). If the overall system consists of a set of loosely coupled agents that act and cooperate with each other in the same environment, a multi-agent system (MAS) is formed. MAS are usually implemented to deal with problems whose complexity overstrains the capabilities of a single agent. For the practical problem solution the agents have to rely on communication capabilities and possibly also on the ability to collaborate, negotiate, delegate responsibility, and trust. These subjects are discussed in detail in literature [cf. e.g. 15].

Multi-agent based simulations (MABS) as used in this paper represent a specific scenario that is modeled and conceptualized as MAS. The involved active and (more or less) independent real world entities are mapped as agents that are embedded in an abstracted, shared and virtual environment. Thus, the foundation of MABS is the modeling and abstraction of the real world scenario, which is basically the definition of a comprehensive environment model and the agents that are situated in this environment. The advantages of MABS are often described by the benefits that result from the conceptualization of a system as an agency. These are: 1. Provision of natural description of a real system, 2. Capture of emergent phenomena in a system and 3. Flexibility [16].

The mapping of individuals from the real-world to an agent based model for simulations can naturally be done by mapping actively acting individuals to agents. Emergent behavior results from the interaction of these individuals. Thus, examples can be found in real social groups as well as in social groups that are simulated by a MABS. Unlike a system that results from a strict mathematical modeling (e. g. with differential equations) with clear overall rules, a MABS may be able to capture group behaviors that are enriched with individual aspects for considerations and reasoning. Those can be differently motivated and reach different decisions for a single situation.

MABS can be considered as a tool for simulations that are able to capture similar and nearly realistic aspects as conventional simulation frameworks, but with a more enriched and meaningful level of detail on the individual agent side [17].

The mentioned flexibility is achieved by the dynamic adaptation of a simulated situation. For such a MABS it is irrelevant how complex a simulation is, as long as the agents are able to couple themselves and interact with an underlying model of the environment. The latter basically reflects the overall complexity of a simulation. For example, a traffic simulation may have to deal with up to 100s, 1,000s or even 100,000s simulated cars. Moreover, agents may be provided with a higher degree of rationality or the ability to learn, which will change their reactions to occurring situations and will again increase complexity. Introductions, descriptions and several exemplary scenarios can be found, e. g., in [18], [19].

In a similar manner MABS is used in large-scale simulation scenarios when testing and validating future smart grid control concepts. These simulations feature a different degree of complexity and may vary in regard to their focus and runtime performance. In most cases it is taken into account when a performative simulation responding to a multitude of inputs is required. In parallel to classical simulations MABS may thereby be limited to simulating certain domain or use-case specific factors and calculating key performance indicators (like voltage or frequency deviations), essential to assess the regarded control.

Even though the degree of complexity, which may be reproduced in MABS is sizeable, real-life implementations feature problems, which may not be considered in a simulation environment. Therefore hybrid test beds exist, which allow for the investigation of such unforeseen event. These feature a combination of real-world devices connected to a simulation environment. The operating data and in return the control commands of these real appliances are exchanged with the simulation, enabling enhanced test scenarios for the control software featuring real delays, and unforeseen events, difficult to model in a software simulation.

In parallel to the described simulation environments, do the current implementations of decentralized grid control solutions have to adapt their behavior and performance in accordance to the existing electricity network. Most of the solutions envisioned do therefore form “virtual power-plants” in which they aggregate decentralized energy converters. Those clusters usually contain a small number of fluctuating electricity production facilities extended by some power generations equipment capable of balancing the unsteady nature of their fellow workmates. The advantage of this type of systems is their controllability in a wide power range, which makes them comparable to conventional power plants. This enables them to provide a predefined amount of energy as agreed upon in earlier negotiated supply agreements between them and external consumers of electrical energy. However, problems in regard to voltage stability may occur due to the dislocated nature of the energy production. An exemplary real-life implementation of such a system is the regenerative model-region Harz [20].

Other projects in the domain of energy management and control are trying to match the demand depending on the availability of energy from renewable energy sources. This technique is called “demand side management”. Most of these implementations are based on smart meters. In connection with time variant electricity rates, incentives are given to use energy-intensive applications in times of low energy

prices, which usually correlate with elevated energy availability [21]. Projects like E-DeMa [22], eTelligence [23], Smart Watts [24] or the Model city of Mannheim [25] do follow this track. These projects do aim at studying the implementation of smart devices in a large infrastructure, focusing on different aspects of electricity generation, consumption and storage while paying special attention to policies and rules. All these projects have one drawback in common, which they share with the majority of alike projects in the international research community - they focus on the electricity domain only. This limitation does not pay tribute to the fact, that the different energy systems are interconnected. An electricity system has to be regarded in the context of the surrounding supply and distribution infrastructure. Not only the gas-, heat- and water network are important counterweights, featuring volatile processes, affecting each other across domain borders, but also meteorological information have to be taken into account. Examples can be found in the linkage of the electricity grid with the heating demand by e.g. night storage heating or the conversion of spare electrical energy to gas by using Power2Gas.

3 An Agent Development Process for Smart Energy Systems

In the following we outline a desirable, systematic development process that covers the development of agents from an initial idea to the deployment in real hardware on site. The main idea herein is that an agent can be moved through different development stages, while the immediate environment of an agent marginally changes. Similar approaches were already presented through methods like Hardware-in-the-loop simulations or Rapid Control Prototyping [26]. In addition, different process models of the Software Engineering can be applied here, as for example the V-Model as it is used by the Germany government [27]. Based on the ideas of these methods, a standardized development cycle for smart energy systems has to include the following project milestones and working packages:

1. **Specification and modelling:** Based on the knowledge of available subsystems, interfaces and information, the desired autonomous software-driven functionalities of an energy entity, as well as the internal and external interactions have to be described by using suitable modelling techniques like UML¹, SysML² or O-MaSE³.
2. **Implementation:** Following the modelling phase, a system, capable of using local information and communicating with other systems in a uniform manner, has to be implemented. This does not require the use of a specific programming language, like C++ or Java, however the communication processes have, at least, to be based on commonly-used data models and standardized interaction methods.
3. **Simulation:** The simulation environment should be able to embed the developed software artefact by providing similar information sources as a real system. Especially when focussing on a decentralized software-use, a simulation framework has to support the required interfaces and data provisioning for the software, in order to

¹ <http://www.uml.org>

² <http://www.sysml.org>

³ <http://macr.cis.ksu.edu/O-MaSE>

- provide or simulate the required independence of a component. For the concurrent simulation of several interacting systems the aspect of autonomy is of great importance, which emphasizes the use of agent based simulations as a precondition and the necessary simulation paradigm. Here, it must be ensured that the overall simulation works at least with a similar temporal behaviour as the real system.
4. **Test bed-application:** in hybrid or entire real-life test-beds the developed software artefact has to be assessed with the hardware used in a later stage. In contrast to simulations, the software is exposed to real-world conditions in terms of available computing capacity and time requirements. Here the developed smart device should be subjected to systematic testing, and so obtain a type of certification for conformity and suitability. If large scale networks have to be tested a hybrid approach, as described in section 2, may be applied. In this type of testbed selected positions are served by real life components, forming an integral part of the environment.
 5. **Deployment to real systems:** The final application in real systems puts high demand on hardware and software requirements in terms of longevity and reliability. For this the hardware used for running the software agent should have a similar life-time as the surrounding electronic system while the software should be maintainable, e. g. through software updates.

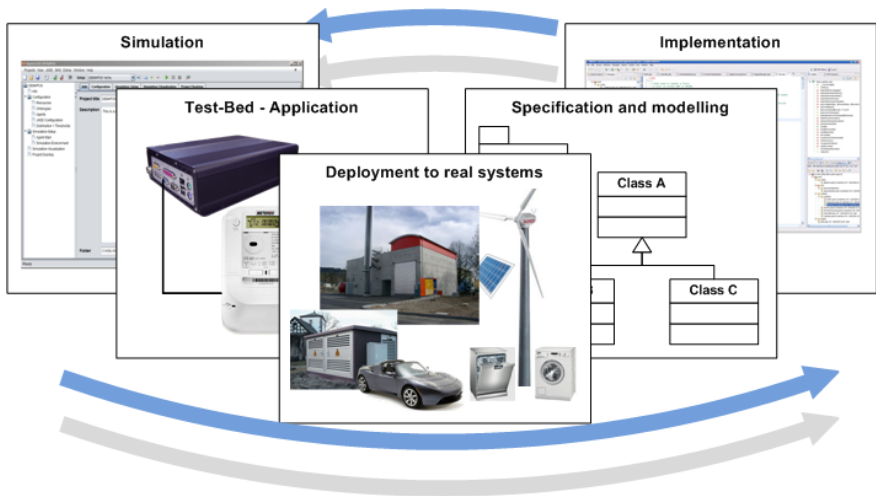


Fig. 1. Desirable development cycle for the systematic development of Energy Agents

As described in the first point, specification and modelling already requires the knowledge about the later on controlled system, its interfaces and its information base. For the entire development process and the concrete development stage we see the actual system in which an Energy Agent is located in as intermediate environment, with accessible (and maybe standardized) interfaces. Since larger energy systems, like power plants, CHP or wind turbines already own local controlling and monitoring processes that are reacting on sensor information, we do not expect that such hard real-time requirements as defined in the domain of automation technology have to be fulfilled by software agents. This is also justified by the fact that, due to

their possible indeterministic behaviour, currently available agent systems appear to be unfavourable in critical real-time systems, demanding fixed execution cycles. Furthermore we see the main application of the agent paradigm in the coordination and harmonization of planning processes in the context of a higher level system, e. g. for the creation of timetables and a coordinated scheduling. In the foreseeable future critical real-time systems - as for example in power plants and other - will still rely on conservative systems located in the domain of the classical automation technology.

Focussing on the real application of agents for smart energy systems shows that a systematic development cycle requires to make assumptions about system boundaries and interfaces that are, for the present time, quite far away from reality. But in the opinion of the author's team such standards are essential for a systematic development of future energy systems. A consistent definition of possible local sub-system boundaries and its interfaces for an autonomous software system called "Energy Agent" would thus help to better understand the roles and functions of such an entity. Furthermore it would be possible - on the long term - to provide comparable scientific results and subsequently allow a sustainable conversion of our energy infrastructure.

4 Definition and Scope of an Energy Agent

After describing its' local environment above, we are now focussing on the definition and the thinkable internal processes and structure of an Energy Agent in a global context. Following [17] and the VDI directive 2653-1 we define thus:

An Energy Agent is a specialized autonomous software system that represents and economically manages the capacitive abilities of the energy consumption, production, conversion and storing processes for a single technical system and that is embedded and thus part of one or more domain specific energy-networks, capable to communicate therein and with external stakeholders.

This definition is based on the considerations of the previous section and with the goal to define a generalized prototype for an Energy Agent that is applicable in various situations and in different energy domains. With the concept of energy domains, we consider not only electrical networks (e. g. Smart Grids) to be more sophisticated in the sense of more dynamic in their resource allocation by using an additional piece of software. Rather, the Energy Agent is supposed to integrate all those energy systems that are subjected to the fundamental first law of thermodynamics, stating that energy can neither be created or destroyed, and which are thus able to store energy or convert it from one form to another. Mathematically written, this means that the sum of consumed and emitted energy of a single system is always equal 0 over time. Thus energy can neither be created nor destroyed. Formula 1 resumes this circumstance.

$$\sum E_i = 0 \quad (1)$$

This theorem of the conservation of energy applies for every system that converts energy from form i to another, like mechanical energy to electrical energy (electricity generator) or chemical energy to thermal energy (heating system). Taking into

account this relationship, we assume all following Energy Agents (and their underlying technical systems) to be individual, but connected by (possibly) hybrid transport networks, each with its' specific form of energy.

In the sense of the first law of thermodynamics and the capacitive abilities of an energy system, the main task for an Energy Agent, connected to a network, is its ability to describe and handle the individual positively or negatively directed (produced/emitted or consumed/obtained) amount of energy over time, maybe in cooperation with its neighbouring entities. This base-requirement leads to the need of a well defined, suitable generic data or option model that allows an Energy Agent (or a later external optimization procedure) to utilize this information.

Derived from this requirement is the question of the necessary and sufficient understanding of time and the system changes over time. On the basis of current implementations of data acquisition systems and for safety reasons it is clear that time has to be discretized and monitored somehow (possibly in equidistant steps) and that it will be dependent on the managed energy usage or production processes. What an ideal step size could be depends basically on the dynamics of the handled amount and type of energy and thus on the physics of the underlying technical system.

Before we discuss a more substantial integration of the energy option model as needed by the Energy Agent and introduced above, we want to divide our definition of an Energy Agent in different levels of sophistication first. These levels describe the local capabilities of onsite software. By doing so, we want to reflect the fact that current systems are usually not equipped with functionalities that could enable complex interactions with their environment, as they are commonly envisioned in the current scientific discussion. For reasons of illustration we define in the following table a coherent set of integration levels for Energy Agents by means of an electricity meter; functionalities and limitations may be transferred to other energy domains as well.

Table 1. Integration level of Energy Agents

Integration Level	Overall Control	Description
IL0	Central	Initial situation: old state of the art from the 80s (e. g. Bakelite ferrite electric meters and newer meters without information exchange)
IL1	Central	Current meter systems: enables information transfer of energy usage, but requires a central data analysis
IL2	Central	Advanced meter systems with predictions: enables information transfer of energy usage with locally aggregated data
IL3	Central & Local	Advanced local controller: Can act on the underlying local system and react autonomously to external signals (e. g. price signals for local optimization)
IL4	Central, Distributed & Local	Advanced local area controller: restricted, but independent local systems that can dynamically build coalitions in order to keep track of optimization goals (e.g. intelligent local power transformers, responsible for one network segment)
IL5	Distributed & Local	Fully distributed control of energy production, distribution and supply

The above classification shows that the consideration of an Energy Agent is advisable if additional control processes are required on a local level. Accordingly, Energy Agents are only necessary from integration level IL1 onwards. Additionally, the table shows that with an increasing integration level a shift to a more decentralized control of energy supply systems is addressed. Integration level IL5 describes thereby a fully autonomous and decentralized energy supply system, based on the most sophisticated Energy Agents. The latter is primarily an extreme, of which we assume that it not will be fully implemented in the near future. Rather we assume that a multitude of future energy supply systems and their parallel development will lead to a concurrent presence of differently sophisticated Energy Agents. This results in another hybrid dimension and thus a further degree of complexity for future energy supply systems. Taking into account the high diversity of the systems to be considered in the context of the Energy Agent definition, different levels of sophistication can already be observed today. Independent of recent scientific solutions for energy-wise autonomous and self-aware systems, we can observe that today's technical reality is mostly located in the integration level IL0 and IL1:

- The broad mass of households is still not equipped with smart-meters that are able to communicate their energy consumption.
- Larger automated plants are designed to follow one or more given set points with their local control systems. The information transfer to central control points is realized by using existing connectivity, while an evaluation of this data is often done offline and only by partly automated processes that were not necessarily designed to optimize energy usage.

Even though these degrees of sophistication may not be seen as use cases for agents at mere sight, the usage becomes clear when regarding integrated systems containing these components. Therefore these basic Integration Levels are needed for simulating current integrated layouts and systems. In order to have a common starting point for discussions and comparisons in the framework of a systematic development approach for future energy systems, the classification of an Energy Agents integration level as discussed briefly in this article will be of great support for reaching a common understanding, especially when refined in future research.

In the above context, another important question lies in the design objective of an Energy Agent or even of a group of Energy Agents, as indicated for the integration levels IL3 to IL5. Here, purely technical or economical goals can be identified and differentiated in the course of developing a holistic system. While technical specifications mostly only describe the parameter set in which a technical system can operate, technical parameters are to be seen as constraints that have to be fulfilled in every case during operation. After fulfilment of such constraints, an optimization can take place. Assuming a kind of cost or income optimization of an Energy Agent, a relation between energy conversion and storage processes and a corresponding accounting system have to be integrated.

A high level quantification of energy flows in units of watt or watt seconds requires an explicit calculation, based on the domain specific form of energy. While for electrical networks voltage and current are the important base units for such calculations, the gas quality and the physical states for pressure, temperature and other

factors are important for gas transportation networks. Thus such detailed domain specific information is also required by an Energy Agent, especially if an energy flow is considered to exceed the boundaries of the underlying single system.

In summary, from the above considerations, this results in the following systemic picture for an Energy Agents and for the needed, aggregated information that have to be gathered, processed and deployed by the internal control processes (see Fig. 2): Running on available or additional hardware, the Energy Agent is connected to the technical system and can partly or fully control and monitor it. By using the domain specific knowledge about the local technical system over time, the agent is able to synthesis an energetic option and cost model (see next section) that can either be used for local limit compliance or for cost optimizations. A further sophistication of an Energy Agent, as it was described with IL3, would allow communication as for example for the processing of external price signals or for coordination processes and negotiations with neighbouring entities, in order to comply with non-local goals.

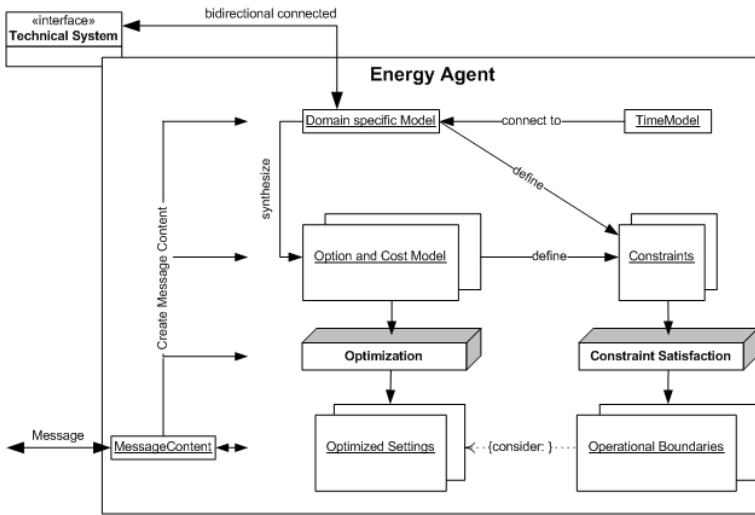


Fig. 2. Structure, interfaces and processes of an Energy Agent of IL3 and higher

5 Option Models: What Energy Agents Can Provide or Not

As stated in the previous section, the description of the time-dependent and quantified energy-related abilities of an Energy Agent or the respective underlying technical system requires summarizing domain-specific information and an abstraction to a high-level, domain-independent system. The necessity for this is derived from the individual optimization goal of Energy Agents at integration level IL3 and above, where a clear relationship between energy price and usage is addressed. Especially, the superior levels IL4 and IL5 require an abstract means of information exchange, in order to communicate their energetic degree of freedom and maybe to negotiate for their concrete energy consumption or production. This particularly applies for the

cross-domain exchange of information and energy, as for example in all relevant intermediate technologies between electricity and gas. Consequently, a unified data model is required that enables the description of individual degrees of freedom and their energy-related abilities of single entities over time.

This means first of all that individual systems (or their managing Energy Agents) need to ‘know’ their technical capabilities, which is trivial for simple consumer such as bulbs, but more difficult for complex systems like cogeneration plants and other. Based on the information, this internal model has to be extended by a temporal behaviour of a single entity. Again, this is simple for a bulb since it can be easily switched on or off, if the control over the bulb is given and if there are no constraints that require the bulb to be permanently switched on. For more complex technical systems, this becomes more difficult since such systems are subject to their current operation state and can not simply be moved to any other operation point without taking into account a multitude of (possibly also external) facts. Therefore, transitive states have to be considered in addition, as for example start-ups, shutdowns, maintenance periods and other.

For the creation of energetic option models, we propose a fundamental two-dimensional subdivision that describes either the typical characteristic of an energy system, while the second dimension classifies a possible temporal behaviour. Here, a possible classification for the energy systems consumption, production and its storage capabilities (C. P. S.) could be as follows:

Table 2. Classification of the characteristics of energy systems

Energy C. P. S.	Examples from different domains
Constant working systems	bulbs, irons, certain types of power plants and other
Task dependent or batch systems	white goods (like washing machines and dishwasher), industrial facilities (start-ups and shutdowns) and other
Repetitive systems	fridges, central heating
Environment dependent systems	wind turbines, photovoltaic plants and other
Dynamic and flexible on-demand-systems	pumped storage plants, gas power plants, gas turbines, compressors, gas storages

A possible temporal behaviour of a single energetic system could range between an uncontrollable usage or production of energy on one side and a fully time-controllable system on the other. Thus, the question is addressed how predictable an energy-related system is and who (or what) controls the system. In addition to manually and thus - from a control perspective - ad hoc operating systems, semi-automated and fully automated systems have to be considered here. Further aspects are the ability to dynamically reduce, increase, pause, shift, switch on or switch off the energy flow at the boundaries of a single system.

The medium-term objective of the above classifications is to build up example cases that can help to create a unified model that is capable of describing all of these cases and that, based on domain-specific information, enables interaction processes

based on a commonly known grammar or language that can be used by every Energy Agent. A necessary extension to that grammar is a corresponding and first internal cost model for actions of Energy Agents and their underlying technical systems.

Based on the above described flexibility aspects of Energy Agents, the question has to be discussed if these internal cost models have to be disclosed, converted and communicated in a standardized manner, in order to enable market oriented control mechanisms and thus the formation of Smart Markets, similar to existing systems featuring automated trading processes. The concrete implementation of a comprehensive, energetic option model is task of our further work and will be presented in a future publication.

6 Conclusion

In the previous sections we have presented a novel approach to and perspective onto the topic of energy transformation and -storage control design and implementation. The suggested Energy Agent as autonomous software system is described as a design concept with several degrees of freedom, covering the majority of use cases in different energy domains. It is supposed to represent and economically manage the abilities of energy related technical systems and shall be capable of communicating with other stakeholders.

For evaluating the functions and capabilities required by the individual Energy Agent several integration levels, referring to the level of evolution, are defined. Six integration levels are differentiated reaching from IL0, denoting entirely local systems incapable of communicating and higher logic, to IL5, representing completely decentralized systems, distributing data acquisition, knowledge and decision onto a variety of entities.

Based on the presented integration levels an analysis of information relevance and flow in a generalized energy management environment is performed. The outcomes are summarized in a schematically model of the Energy Agent illustrating the most important steps regarding information acquisition, processing and deployment while complying with given constraints in regards to the encountered technical and economical context.

Derived from this model an option model is outlined, supporting the classification of energy usage patterns, allowing for an efficient provision and processing of vital information.

Furthermore a systematic development approach covering the core functions of the Energy Agent was introduced. This approach is based on strategies developed and successfully deployed in the fields of automation technology. It comprises the following steps: specification and modelling, implementation, simulation, test bed-application and finally the deployment to real systems. This work is the offspring of a scientific cooperation, in which a decentralized energy- and load- management system was linked to an agent based multi domain simulation. In the course of the study several pitfalls and shortcomings in regard to communication and data management processes in the energy domain were identified. The subsequently envisioned Energy

Agent is supposed to serve as a basis for further development and discussion in the domain of decentralized energy management. Questions in regard to standardization shall be raised with a special focus on communication and system design. While systematically specifying the Energy Agent in further iteration cycles, generalizable requirements shall be derived, supporting the development of future energy control systems.

Until now the described Energy Agent lacks a proper definition of stakeholders and their corresponding user roles and privacy requirements. These factors shall be taken in account when implementing the first exemplarily release in the near future.

References

- [1] Ilic, M.D., Allen, H., Chapman, W., King, C.A., Lang, J.H., Litvinov, E.: Preventing Future blackouts by Means of Enhanced electric Power Systems Control From Complexity to Order. *Proceedings of the IEEE* (11) (2005)
- [2] Setiawan, E.A.: *Concept and Controllability of Virtual Power Plant*. Kassel University Press (2007)
- [3] Nestle, D., Ringelstein, J.: Integration of DER into Distribution Grid Operation and Decentralized Energy Management. In: *Smart Grids Europe 2009, Barcelona* (March 19, 2009)
- [4] Daneels, A., Salter, W.: What is SCADA? In: *International Conference on Accelerator and Large Experimental Physics Control Systems, Trieste* (1999)
- [5] Huhns, M.N., Stephens, L.M.: *Multiagent Systems and Societies of Agents*. In: *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT press (1999)
- [6] Jennings, N.R., Wooldridge, M.: *Applications of intelligent agents*. In: *Agent Technology*. Springer (1998)
- [7] Keilar, N.: *Concept to Integrate Distributed Generation into the German Power Grid Intended for Multi-Agent System Realisation*. Thesis, http://www.hsu-hh.de/evhs/index_WCPxnDvauo2WCG70.html (version: April 2013)
- [8] Li, H., Sun, H., Wen, J., Cheng, S., He, H.: A Fully Decentralized Multi-Agent System for Intelligent Restoration of Power Distribution Network Incorporating Distributed Generations. *IEEE Comput. Intell. Mag.* 7, 66–76 (2012)
- [9] Kok, K., Warmer, C., Kamphuis, R., Mellstrand, P., Gustavsson, R.: Distributed Control in the Electricity Infrastructure. In: *Proceedings of the International Conference on Future Power Systems* (2005)
- [10] Lehnhoff, S.: *Dezentrales vernetztes Energiemanagement - Ein Ansatz auf Basis eines verteilten Realzeit-Multiagentensystems*. Vieweg + Teubner (2010)
- [11] Platt, G.: The Decentralised Control of Electricity Networks- Intelligent and Self-Healing Systems. In: *Grid-Interop-Forum 2007*, pp. 1–6 (2007)
- [12] Linnenberg, T., Wior, I., Schreiber, S., Fay, A.: A Market-Based Multi-Agent System for Decentralized Power and Grid Control. In: *Proceedings of IEEE-ETFA* (2011), doi:10.1109/ETFA.2011.6059126
- [13] Derksen, C., Branki, C., Unland, R.: A framework for agent-based simulations of hybrid energy infrastructures. In: Ganzha, M., Macias-zek, L.A., Paprzycki, M. (eds.) *FedCSIS* (2012)

- [14] Sudeikat, J., Braubach, L., Pokahr, A., Lamersdorf, W., Renz, W.: Validation of BDI agents. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) PROMAS 2006. LNCS (LNAI), vol. 4411, pp. 185–200. Springer, Heidelberg (2007)
- [15] Wooldridge, M.: An Introduction to MultiAgent Systems, 2nd edn. Wiley & Sons (July 2009)
- [16] Moss, S., Davidsson, P. (eds.): MABS 2000. LNCS (LNAI), vol. 1979. Springer, Heidelberg (2001), <http://books.google.de/books?id=15eRqAZh0JgC>
- [17] Davidsson, P.: Multi agent based simulation: Beyond social simulation. In: Moss, S., Davidsson, P. (eds.) MABS 2000. LNCS (LNAI), vol. 1979, pp. 97–107. Springer, Heidelberg (2001)
- [18] Bonabeau, E.: Agent-based modeling: methods and techniques for simulating human systems. Proceedings of the National Academy of Sciences of the United States of America 99(suppl. 3), 7280–7287 (2002)
- [19] Drogoul, A., Vanbergue, D., Meurisse, T.: Multi-agent based simulation: Where are the agents? In: Sichman, J.S., Bousquet, F., Davidsson, P. (eds.) MABS 2002. LNCS (LNAI), vol. 2581, pp. 1–15. Springer, Heidelberg (2003)
- [20] RegenerativeKraftwerke Harz: REG MOD HARZ. Regenerative Modellregion Harz Homepage, <https://www.regmodharz.de/das-projekt/> (version: April 2013)
- [21] Faruqui, A., Sergici, S.: Household Response to Dynamic Pricing of Electricity—A Survey of the Experimental Evidence. Journal of Regulatory Economics (2013), http://www.hks.harvard.edu/hepg/Papers/2009/The%20Power%20of%20Experimentation%20_01-11-09_.pdf (version: April 2013)
- [22] Laskowski, M.: E-DeMa – development and demonstration of locally networked energy systems to the E-Energy marketplace of the future. E-DeMa Homepage, <http://www.e-dema.com/en/index.html> (version: April 2013)
- [23] AG, EWE: eTelligence. eTelligence Homepage, <http://www.etelligence.de/etelligence.html> (version: April 2013)
- [24] Utilicount GmbH & Co. KG: Smart Watts - Startseite. Smart Watts Homepage, <http://www.smartwatts.de/home.html> (version: April 2013)
- [25] MVV Energie AG: moma – The power supply system gets intelligent. Modellstadt Mannheim Homepage., <http://www.modellstadt-mannheim.de/moma/web/en/home/index.html> (version: April 2013)
- [26] Boehm, B.: Guidelines for verifying and validating software requirements and design specifications. In: EURO. IFIP, vol. 79, pp. 711–719 (1979)
- [27] Abel, D., Bollig, A.: Rapid Control Prototyping - Methods and Application. 1. Aufl. Springer, Berlin (2006) ISBN 3-540-29524-0

Efficient Mechanism for Aggregate Demand Prediction in the Smart Grid

Péter Egri¹ and József Váncza^{1,2}

¹ Fraunhofer Project Center for Production Management and Informatics
Institute for Computer Science and Control

Hungarian Academy of Sciences

Kende u. 13-17, 1111 Budapest, Hungary

² Department of Manufacturing Science and Technology

Budapest University of Technology and Economics

H-1111 Budapest, Egy József u. 1., Hungary

{egri,vancza}@sztaki.mta.hu

Abstract. This paper studies an aggregate demand prediction problem relevant in smart grids. In our model, an aggregator agent is responsible for eliciting the demand forecasts of a number of self-interested home agents and purchasing electricity for them. Forecasts are given in form of probability distributions, and generating them incurs costs proportional to their precision. The paper presents a novel scoring rule based mechanism which not only makes the agents interested in reporting truthfully, but also inspires them to achieve the socially optimal forecast precision. Hence, the aggregator agent is then able to optimise the total expected cost of electricity supply. Therefore the mechanism becomes efficient, contrarily to prior works in this field. Empirical studies show that it is beneficial to join to the mechanism compared to purchasing electricity directly from the market, even if the mechanism consists only of a few agents.

Keywords: Smart grid, distributed optimisation, information aggregation, mechanism design, scoring rules.

1 Introduction

In the *energy sector*, supplies are meticulously provided so as to meet projected loads. On the one hand, when actual demand exceeds supply, the response is to use increasingly expensive energy generation sources, and, in the last resort, to decrease load (e.g., through brownouts and blackouts). On the other hand, overproduction of energy incurs severe extra costs, too. The resolution relies heavily on multiplexing a set of loads in order to smooth the aggregate demand which is formed from information provided by local individual users of energy [1]. This coupling of energy and information flows is the basis of *smart grids*.

The paper discusses an aggregate demand prediction model where self-interested *home agents* can produce electricity demand forecasts at a certain cost.

The independent agents report the forecasts to an *aggregator agent* that is responsible for meeting the actual total demand. While the system as a whole could operate most efficiently if the home agents would invest into making precise enough forecasts and share this information truthfully with the aggregator, in a system of self-interested agents there is no warranty for such a behaviour. Why would not take any home agent a free rider's stance and neglect forecasting? And why would it report its unbiased private forecast when it wants to make sure its demand will be met by the electricity supply? How could agents be made interested in putting effort into generating their own, sufficiently precise demand forecasts? What an incentive system could warrant truthful communication of these forecast information? In general, can any coordination mechanism facilitate the better (i.e., cheaper, more profitable, more stable) behaviour of the system as a whole than the simple aggregation of the individual operations? Why would it pay off for any agent to participate in such a system instead of going for the satisfaction of its own demand directly?

This problem setting is relevant in any domain where demand and supply has to be matched by autonomous partners who have asymmetric information and responsibilities, and where the satisfaction of aggregate, uncertain demand may incur lower costs (or higher profits) than meeting individual demands apiece. For instance, it may mitigate risks implied by market volatility as well as decrease total production and logistics costs in *supply networks* where a supplier is serving different retailers of the same product [2].

We investigate the above issues by means of the apparatus of *mechanism design*. Mechanism design, also considered inverse game theory, has a specific engineering perspective: it applies the model of *non-cooperative games* with agents having incomplete information, and investigates how the private information influencing the other agents' utilities can be elicited [3]. Thanks to this generic approach, beyond elaborating a complex—communication, decision and financial transaction—mechanism we can also prove some key properties of this inherently distributed system that are prerequisites of any real application.

The remainder of the paper is organised as follows. In the next section we review related works, then in Section 3 present the demand prediction and decision making model, and derive its optimal solution in a cooperative, social welfare maximising setting. In Section 4 the proposed mechanism is presented together with the proofs of its main properties in the non-cooperative case. Numerical illustrations of the system behaviour is shown in Section 5. Finally, the paper concludes in Section 6.

2 Literature Review

The *information elicitation* or *prediction mechanism design* problem, which has recently come into the focus also of the multiagent research, consists of several agents with some private information about the probability of some stochastic future event, and a centre whose goal is to obtain and aggregate the dispersed information [4,5]. There are two different kinds of such models: the *peer prediction systems* and the *prediction markets*. In the former one, the outcome (event)

is subjective or unmeasurable, therefore it cannot be used as a basis for evaluating the forecasts posteriorly. In such cases, the predictions of the agents can be compared to other agents' forecasts only.

In prediction markets however, which are in our main focus from now on, there is always a clear, objective outcome. Such problems can be handled by applying the so-called *strictly proper scoring rules* that we briefly define here. Let us assume a set D of possible events, and \mathcal{P} , a class of probability measures over them. A scoring rule $S : \mathcal{P} \times D \rightarrow \mathbb{R}$ is called strictly proper, if whenever an event $\xi \in D$ is drawn from the distribution $\theta \in \mathcal{P}$, then for any other $\hat{\theta} \neq \theta : \mathbb{E}_\theta[S(\theta, \xi)] < \mathbb{E}_\theta[S(\hat{\theta}, \xi)]$. With other words, the score can be minimised (in expectation), if it is parametrised with the real distribution of the stochastic event.¹ Well-studied examples for strictly proper scoring rules are the *quadratic*, the *spherical* and the *logarithmic* rules, see e.g., [4]. Applying such rules to the information elicitation problem is straightforward: if the agent with the private information is penalized proportionally to a proper score, it becomes interested in creating and providing as good a forecast as possible.

Note that it is implicitly assumed in these models that the forecast can be generated free of charge; when generating or improving the forecast involves some cost, truthful mechanisms may not exist in general [4]. However, there exist some cases when costly forecasts can be successfully included into the model. For example, in [6] several agents can provide forecasts for the same stochastic variable at different costs, and the authors present a two-stage mechanism including a reverse second-price auction for solving the information elicitation problem in this case.

Apart from the costly forecast generation, there are several other extensions of the information elicitation problem. In some models, the agents have interests in the decision of the centre, therefore they might disclose false information in order to manipulate the decision maker. Such situation is considered e.g., in [2], where the logistic decisions of a supplier can cause shortages at the retailers, which affects their profits. Further difficulties occur, when the objective function of the agents are not known exactly, or when the agents can manipulate the outcome and therefore influence the evaluation of their reported forecasts.

Information elicitation problems appear in smart grids, which intend to combine modern communication technology with the electricity network. In order to level fluctuations and optimise the generation, distribution and utilisation of electricity, the recent advances in hardware technology should be supplemented with novel software support of automation and control, as well as business models [7]. The so-called "smart meters" for example, that measure not only the electricity usage, but also record the time of the consumption, enable the introduction of time-differentiated pricing. Exploiting the price-responsive demand, this instrument can be applied as a tool for aligning the demand with the supply of fluctuating renewable energy sources (e.g., wind, solar) in the smart grids. The bidirectional communication also enables that utility companies can collect

¹ Note that in contrast to the usual notation, for convenience, we minimize the score.

and analyse more detailed and precise information generated by agent-based forecasting at household level [8].

In [9] an aggregate demand prediction model is presented for the smart grid, where an aggregator agent elicits consumption forecasts from the home agents and purchases electricity for them. The authors present a scoring rule based mechanism that fairly distributes the savings among the agents, and prove its individual rationality, incentive compatibility and ex ante weak budget balance properties (for formal definitions, see later). However, incentive compatibility in that paper relates only to the truthful reporting of forecasts, their precision is usually not globally optimal, therefore the mechanism is not efficient. In addition, the mechanism has to artificially limit the accepted precision from the agents, thus even the optimal solution could be excluded.

A structurally similar model for supply networks is presented in [10]. In that paper a supplier collects demand forecasts from retailers, and provides a Vendor Managed Inventory (VMI) service for them. The model completely disregards the costs of forecasting, and presents a mechanism that is incentive compatible and efficient, but individual rationality is not discussed.

In general, the operations research literature provides several analytical models for optimisation of decision making in the above setting. The model in this paper applies a specialised version of the single-period stochastic lot-sizing problem, the so-called *newsvendor* model [11]. Since this assumes already given forecasts, the costly forecasting should be included into the model by comparing the cost and the benefit of generating an estimate with a given precision. For this problem, the *value of information* (VOI) approach can be applied [12].

It is not unprecedented in the literature to apply coordination contracts in the electricity industry originally developed for supply chain inventory management. In [13] different contractual forms and market structures are investigated for dampening the *double marginalisation* effect, i.e., when the rational decisions of self-interested decision makers lead to a non-Pareto optimal equilibrium. The problem is illustrated through a case study of the Spanish electricity market, where the demand can be fulfilled either from the futures (forward) market in advance, or promptly from the spot (balancing) market. In that paper, the demand is not stochastic but price-dependent, therefore the price can be used to balance demand with supply.

A related study can be found in [14], which focuses on the energy resources instead of consumption. The authors present a model and incentive mechanism for aggregating electricity production from distributed energy resources. In their model the estimation is characterised by the expected value only, which is later compared to the realised production, therefore no scoring rules are required.

In this paper we consider a similar model as in [9], but contrarily, our mechanism is efficient, does not bind the accepted precision, and inspires the home agents to generate the optimally precise forecasts. The presented mechanism is based on the scoring rule introduced in [10], but it is extended to consider the costly forecast generation. Furthermore, we prove additional properties of the mechanism in the discussed model like individual rationality and ex ante strong

budget balance. Since the proposed transfer function of agent i does not depend on other agents' forecasts or demands, it evaluates only the correctness of agent i 's estimation, in this sense it can be considered fair, and provides the privacy of the consumption data.

3 Model

We assume n home agents and an aggregator agent who collects forecasts about the future consumption of the homes, and purchases electricity for them (see Fig. 1). Each day is divided into a number of periods, and for each period a home agent can generate demand forecast at a cost proportional to the forecast precision. From now on, we consider only one period, since the problems for different periods are independent from each other.

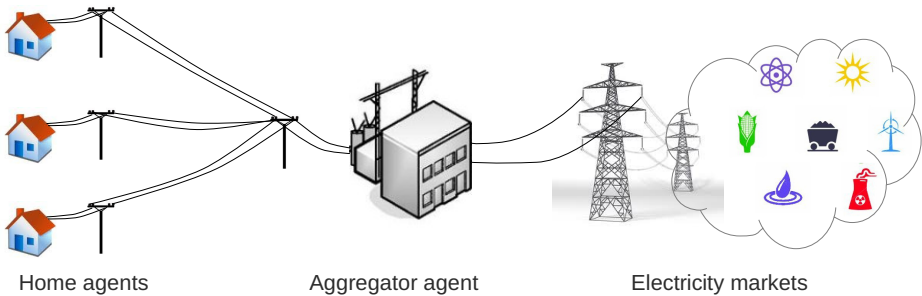


Fig. 1. Electricity network

We assume that the forecast of agent $i \in [1, n]$ is represented by a *normal distribution* with mean m_i and standard deviation σ_i . This forecast can be generated at price $p(\sigma_i) = \alpha/\sigma_i^2$, where $1/\sigma_i^2$ is the so-called *precision* of the forecast, and $\alpha > 0$ is a constant. Note that we assume that the price function is the same for all agents.

It is also assumed that the consumptions at the homes are independent from each other, therefore the aggregate forecast of the total consumption will also be normally distributed with mean $m = \sum_{i=1}^n m_i$ and $\sigma = \sqrt{\sum_{i=1}^n \sigma_i^2}$.

The aggregator agent decides about the purchase quantity q that is bought on the *forward market* at price c . During the period, agent i consumes ξ_i electricity, and if the total realized consumption $\xi = \sum_{i=1}^n \xi_i$ is less or equal than the quantity bought on the forward market, the aggregator can provide the necessary electricity. However, if $\xi > q$, the aggregator has to buy additional electricity on the *balancing market* at *buy price* b . On the other hand, if $\xi < q$, the surplus can be sold there at *sell price* s . It is natural to assume the following relation between the different prices: $b > c > s$.

In the following two subsections we study the social welfare maximising solution in the centralised model considering cooperative agents. The resulted decision problem has two stages: in the first stage the optimal forecast precisions, while in the second stage the optimal purchase quantity is computed. We solve the problem in a *backward induction* manner: firstly, we determine the optimal q assuming a given forecast with a special form of the newsvendor model, then we derive the optimal forecast precision.

3.1 Optimal Purchase Quantity in the Cooperative Case

In this subsection we consider the normally distributed total consumption $\xi \sim \mathcal{N}(m, \sigma)$, and denote its probability and cumulative distribution functions with ϕ and Φ , respectively. If the aggregator purchases quantity q , its resulted revenue will be $-cq - b \max(\xi - q, 0) + s \max(q - \xi, 0)$, i.e., the payment for the electricity on the forward market and the (negative or positive) payment of matching supply and demand on the balancing market. The expected value of this, which we call *valuation*, can be expressed in the following form:

$$\begin{aligned}
 v(q) &= -cq - b\mathbb{E}[\max(\xi - q, 0)] + s\mathbb{E}[\max(q - \xi, 0)] \\
 &= -cq - b \int_q^\infty (x - q)\phi(x)dx + s \int_{-\infty}^q (q - x)\phi(x)dx \\
 &= -cq - b \left(\int_q^\infty x\phi(x)dx - q \int_q^\infty \phi(x)dx \right) \\
 &\quad + s \left(q \int_{-\infty}^q \phi(x)dx - \int_{-\infty}^q x\phi(x)dx \right) \\
 &= -cq - b \left(m - \int_{-\infty}^q x\phi(x)dx - q(1 - \Phi(q)) \right) \\
 &\quad + s \left(q\Phi(q) - \int_{-\infty}^q x\phi(x)dx \right) \\
 &= -cq + b(q - m) + (b - s) \left(\int_{-\infty}^q x\phi(x)dx - q\Phi(q) \right). \tag{1}
 \end{aligned}$$

Since the valuation function is concave ($v''(q) = -(b-s)\phi(q) \leq 0$), the optimal q^* can be determined by the first derivative test

$$v'(q^*) = b - c - (b - s)\Phi(q^*) = 0, \tag{2}$$

which results in

$$q^* = \Phi^{-1} \left(\frac{b - c}{b - s} \right) = m + \sigma\sqrt{2} \operatorname{erf}^{-1} \left(\frac{b - 2c + s}{b - s} \right), \tag{3}$$

where

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt, \tag{4}$$

is the *Gauss error function* and $\text{erf}^{-1}(x)$ denotes its inverse. The valuation using the *optimal purchase quantity* can then be expressed by substituting (3) into (1) as

$$v^*(\sigma) := v(q^*) = -cm - \sigma K_{cbs} , \tag{5}$$

where

$$K_{cbs} = \frac{(b-s)e^{-\left(\text{erf}^{-1}\left(\frac{b-2c+s}{b-s}\right)\right)^2}}{\sqrt{2\pi}} . \tag{6}$$

Note that K_{cbs} depends only on the cost parameters c , b and s .

3.2 Optimal Forecast Precision in the Cooperative Case

We now examine the optimal forecast precisions, or equivalently, the optimal standard deviations. Since we assume that the price function of the forecast generation is the same at every agent ($p(\sigma_i) = \alpha/\sigma_i^2$), it follows that in the optimal solution each σ_i is the same ($\sigma_i = \sigma_j, \forall i, j \in [1, n]$) and therefore $\sigma = \sqrt{\sum_{i=1}^n \sigma_i^2} = \sqrt{n}\sigma_i$.

The *utility* will be the valuation minus the forecasting price of the n agents:

$$U(\sigma_i) = v^*(\sigma) - np(\sigma_i) = -cm - \sqrt{n}\sigma_i K_{cbs} - n\frac{\alpha}{\sigma_i^2} . \tag{7}$$

This utility function is concave too ($U''(\sigma_i) = -6n\alpha/(\sigma_i)^4 \leq 0$), therefore the optimal standard deviation can be determined by

$$U'(\sigma_i^*) = 2n\alpha/(\sigma_i^*)^3 - \sqrt{n}K_{cbs} = 0 , \tag{8}$$

which results in

$$\sigma_i^* = \sqrt[3]{\frac{2\sqrt{n}\alpha}{K_{cbs}}} . \tag{9}$$

The optimal forecast precision in function of the number of the home agents is illustrated on Fig. 2.

4 Non-cooperative Mechanism

In this section we consider a non-cooperative setting, where each agent maximises its own utility. A home agent has two decisions: about the precision of the generated forecast (σ_i), and the values of the reported forecast ($\hat{m}_i, \hat{\sigma}_i$), while the aggregator decides about the purchased quantity (q). The mechanism in this case is characterised by a *transfer function* or payment $t_i(\hat{m}_i, \hat{\sigma}_i, \xi_i)$ that has to be paid by agent i to the aggregator after the consumption ξ_i is realised. The transfer function should depend only on commonly known parameters and independent from the privately known m_i and σ_i .

We are looking for a mechanism that is realized by such a transfer function that fulfils the following four key properties:

- It has *incentive compatibility* that implies in this case two things: (i) the home agents create optimally precise forecasts, and (ii) they report the forecasts to the aggregator truthfully, if they want to maximise their utility.
- The mechanism is *efficient*, i.e., the aggregator agent purchases the optimal quantity of electricity from the market.
- The mechanism is *individually rational*, i.e., the utility of the home agents are not less than their utility without the mechanism (when purchasing directly from the market).
- Finally, it is *budget balanced* meaning that the mechanism does not run into deficit or surplus, or in other words, the utility of the aggregator is zero.

We assume that the market prices c , b and s are common knowledge, since the home agents can buy directly from the market if they do not join the mechanism. We suggest the same transfer function that we developed for coordinating supply chains, that consists of two terms: (i) the payment for the purchased electricity, and (ii) a penalty for the forecast error, based on a scoring rule [10]:

$$t_i(\hat{m}_i, \hat{\sigma}_i, \xi_i) = c\xi_i + \gamma \left(\frac{(\xi_i - \hat{m}_i)^2}{\hat{\sigma}_i} + \hat{\sigma}_i \right), \quad (10)$$

where \hat{m}_i and $\hat{\sigma}_i$ are the communicated forecast of agent i , ξ_i is its realised consumption, while γ is a positive constant. As we shall soon see, in order to achieve the four required properties stated above, γ cannot be arbitrary, but it should assume a specific value.

It is worth noting that the payment (10) is very similar to the classical *logarithmic scoring rule*, which can be determined as the negated logarithm of the probability density function of the communicated forecast, i.e., in case of the normal distribution, it is the following

$$\frac{\ln 2\pi}{2} + \left(\frac{(\xi_i - \hat{m}_i)^2}{2\hat{\sigma}_i^2} + \ln \hat{\sigma}_i \right). \quad (11)$$

It is well known that a positive affine transformation of a strictly proper scoring rule remains strictly proper [4], but in spite of the similarities, these two scoring rules are not affine transformations of each other.

In what follows we examine one by one whether and how the non-cooperative mechanism exhibits the basic properties required above.

4.1 Incentive Compatibility

In the first phase the home agents generate forecasts m_i and σ_i , next, they report the forecasts to the aggregator agent in the centre. Let us first examine the latter phase with generated forecast (m_i, σ_i) , reported forecast $(\hat{m}_i, \hat{\sigma}_i)$, and realized demand ξ_i . Note that at this point since $p(\sigma_i)$ has been already invested in the forecast, we consider only the expected payment here.

Theorem 1. *The unique optimal solution for minimising the expected payment is $\hat{m}_i = m_i$ and $\hat{\sigma}_i = \sigma_i$, therefore the home agents are inspired to report the forecasts truthfully.*

Proof. The expected payment is

$$\begin{aligned}
 \mathbb{E}[t_i(\hat{m}_i, \hat{\sigma}_i, \xi_i)] &= c\mathbb{E}[\xi_i] + \gamma\mathbb{E}\left[\frac{(\xi_i - \hat{m}_i)^2}{\hat{\sigma}_i} + \hat{\sigma}_i\right] \\
 &= cm_i + \gamma\mathbb{E}\left[\frac{\xi_i^2 + \hat{m}_i^2 - 2\hat{m}_i\xi_i}{\hat{\sigma}_i} + \hat{\sigma}_i\right] \\
 &= cm_i + \gamma\frac{\mathbb{E}[\xi_i^2] + \hat{m}_i^2 - 2\hat{m}_i\mathbb{E}[\xi_i]}{\hat{\sigma}_i} + \hat{\sigma}_i \\
 &= cm_i + \gamma\left(\frac{m_i^2 + \sigma_i^2 + \hat{m}_i^2 - 2\hat{m}_im_i}{\hat{\sigma}_i} + \hat{\sigma}_i\right), \tag{12}
 \end{aligned}$$

where we have applied the identity $\mathbb{E}[\xi_i^2] = m_i^2 + \sigma_i^2$. The partial derivative of the expected payment by \hat{m}_i is

$$\frac{\partial\mathbb{E}[t_i(\hat{m}_i, \hat{\sigma}_i, \xi_i)]}{\partial\hat{m}_i} = \gamma\left(\frac{2\hat{m}_i - 2m_i}{\hat{\sigma}_i}\right), \tag{13}$$

which equals zero iff $\hat{m}_i = m_i$, independently from the value of $\hat{\sigma}_i$. This yields the minimum, since the expected payment is convex in \hat{m}_i :

$$\frac{\partial^2\mathbb{E}[t_i(\hat{m}_i, \hat{\sigma}_i, \xi_i)]}{\partial\hat{m}_i^2} = \gamma\left(\frac{2}{\hat{\sigma}_i}\right) \geq 0. \tag{14}$$

For calculating the other partial derivative, we already exploit that $\hat{m}_i = m_i$:

$$\frac{\partial\mathbb{E}[t_i(m_i, \hat{\sigma}_i, \xi_i)]}{\partial\hat{\sigma}_i} = \gamma\left(-\frac{\sigma_i^2}{\hat{\sigma}_i^2} + 1\right), \tag{15}$$

which equals zero iff $\hat{\sigma}_i = \sigma_i$, which is the minimum, since the expected payment is convex also in $\hat{\sigma}_i$:

$$\frac{\partial^2\mathbb{E}[t_i(m_i, \hat{\sigma}_i, \xi_i)]}{\partial\hat{\sigma}_i^2} = \gamma\frac{\sigma_i^2}{\hat{\sigma}_i^3} \geq 0. \tag{16}$$

□

Let us now examine the first phase knowing that later the forecasts will be reported truthfully. In this case, the expected payment can be derived from (12): $\mathbb{E}[t_i(m_i, \sigma_i, \xi_i)] = cm_i + 2\gamma\sigma_i$, thus the utility of agent i becomes:

$$U_i(\sigma_i) = -\mathbb{E}[t_i(m_i, \sigma_i, \xi_i)] - p(\sigma_i) = -cm_i - 2\gamma\sigma_i - \frac{\alpha}{\sigma_i^2}. \tag{17}$$

Since the utility is concave in σ_i ($U_i''(\sigma_i) = -6\alpha/(\sigma_i)^4 \leq 0$), the optimal standard deviation can be determined by $U_i'(\sigma_i^*) = -2\gamma + 2\alpha/(\sigma_i^*)^3 = 0$, which yields

$$\sigma_i^* = \sqrt[3]{\frac{\alpha}{\gamma}}, \tag{18}$$

therefore using $\gamma = \frac{K_{chs}}{2\sqrt{n}}$ results in the the optimal forecast precision of (9).

4.2 Efficiency

The utility of the aggregator will be the collected transfer payments minus the price of the electricity bought, plus the price of the electricity eventually sold—this last two items are summed up by the valuation (1), i.e.,

$$U_a(q) = \mathbb{E} \left[\sum_{i=1}^n t_i(m_i, \sigma_i, \xi_i) \right] + v(q). \quad (19)$$

Since the first term is independent from the decision variable q , thus the aggregator intends to maximise the valuation that results in the optimal quantity derived in (3). Hence, the system as a whole meets the demand of home agents in expectation at the lowest possible total price of electricity.

4.3 Individual Rationality

Without the mechanism, home agent i would generate its forecast and buy the electricity directly from the forward and balancing markets. Now we investigate if any home agent i could be better off without joining to the mechanism. Let q_i denote the quantity bought on the forward market, then analogously to the derivation of Section 3.1, its revenue become $-cq_i - b \max(\xi_i - q_i, 0) + s \max(q_i - \xi_i, 0)$, and its expected value

$$v_i(q_i) = -cq_i + b(q_i - m_i) + (b - s) \left(\int_{-\infty}^{q_i} x \phi_i(x) dx - q_i \Phi_i(q_i) \right), \quad (20)$$

where ϕ_i and Φ_i are the probability and cumulative distribution functions of ξ_i , respectively. This yields an optimal purchase quantity

$$q_i^* = m_i + \sigma_i \sqrt{2} \operatorname{erf}^{-1} \left(\frac{b - 2c + s}{b - s} \right), \quad (21)$$

and

$$v_i^*(\sigma_i) := v_i(q_i^*) = -cm_i - \sigma_i K_{cbs}. \quad (22)$$

Then the optimal forecast precision can be derived analogously to Section 3.2 using

$$\bar{U}_i(\sigma_i) = v_i^*(\sigma_i) - p(\sigma_i) = -cm_i - \sigma_i K_{cbs} - \frac{\alpha}{\sigma_i^2} : \quad (23)$$

$$\bar{\sigma}_i^* = \sqrt[3]{\frac{2\alpha}{K_{cbs}}}. \quad (24)$$

All in all, the utility of agent i without the mechanism is

$$\begin{aligned} \bar{U}_i(\bar{\sigma}_i^*) &= v_i^*(\bar{\sigma}_i^*) - p(\bar{\sigma}_i^*) = -cm_i - \bar{\sigma}_i^* K_{cbs} - p(\bar{\sigma}_i^*) \\ &= -cm_i - \frac{3}{\sqrt[3]{4}} \sqrt[3]{\alpha K_{cbs}^2}. \end{aligned} \quad (25)$$

However, by using the mechanism (9) and (17) are valid, thus

$$\begin{aligned}
 U_i(\sigma_i^*) &= -\mathbb{E}[t_i(m_i, \sigma_i^*, \xi_i)] - p(\sigma_i^*) \\
 &= -cm_i - \frac{3}{\sqrt[3]{4n}} \sqrt[3]{\alpha K_{cbs}^2},
 \end{aligned}
 \tag{26}$$

and comparing (25) and (26) shows that $U_i(\sigma_i^*) > \bar{U}_i(\bar{\sigma}_i^*)$ (if $n > 1$), i.e., the utility using the mechanism is always greater than without the mechanism. Hence, the home agents have an incentive to use the service of the demand aggregator mechanism when meeting their individual demand for electricity.

4.4 Budget Balance

The utility of the aggregator can be computed substituting (3), (5), (9) into (19):

$$\begin{aligned}
 U_a(q^*) &= \mathbb{E} \left[\sum_{i=1}^n t_i(m_i, \sigma_i^*, \xi_i) \right] + v^*(\sigma^*) \\
 &= n\mathbb{E}[t_i(m_i, \sigma_i^*, \xi_i)] - cm - \sqrt{n}\sigma_i^* K_{cbs} \\
 &= n(cm_i + 2\gamma\sigma_i^*) - cm - \sqrt{n}\sigma_i^* K_{cbs} \\
 &= cm + 2n\gamma\sigma_i^* - cm - \sqrt{n}\sigma_i^* K_{cbs} = 2n\gamma\sigma_i^* - \sqrt{n}\sigma_i^* K_{cbs} \\
 &= 2n \frac{K_{cbs}}{2\sqrt{n}} \sqrt[3]{\frac{2\sqrt{n}\alpha}{K_{cbs}}} - \sqrt{n} \sqrt[3]{\frac{2\sqrt{n}\alpha}{K_{cbs}}} K_{cbs} = 0,
 \end{aligned}
 \tag{27}$$

thus the mechanism is ex ante (in expectation) budget balanced. In other words, no payments or debts are accumulated at the aggregator agent.

5 Computational Study

In this section we illustrate the properties of the mechanism on a specific numerical example and simulation runs. We apply the same experimental set-up as in [9], i.e., we set $c = 100$, $b = 170$, $s = 50$ and $\alpha = 20$. Fig. 2 shows the optimal forecast precision as the number of home agents increases. It can be observed that as more and more agents join the mechanism, the required precision—thus the forecasting cost—considerably decreases.

Fig. 3 plots the expected and the simulated average utility of the home agents, where the expected demand (m_i) for each agent is uniformly distributed in the interval $[30, 50]$. Each value is an average of 1000 simulation runs, while the expected cost curve is given by (26).

The previous two figures point out that the largest marginal gain joining the mechanism is achieved when there are only few agents in the mechanism already. This means that although joining a larger coalition is always better than joining a smaller, but only slightly. In other words, a lot of independent mechanisms with relatively small number of homes are almost as economical as a single mechanism with all the agents. It is an interesting property that a mechanism

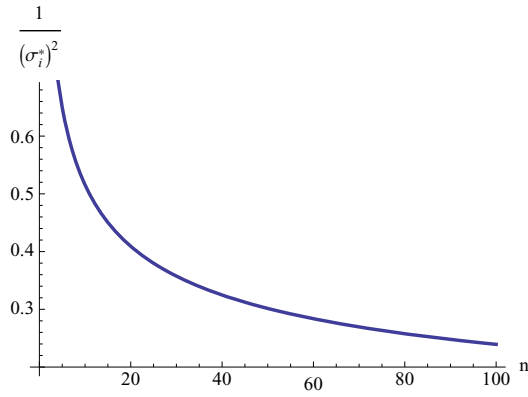


Fig. 2. Required forecast precision

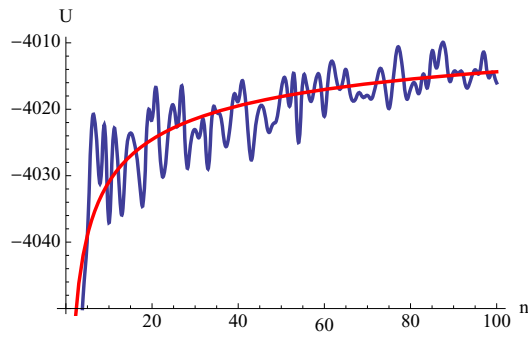


Fig. 3. Average utility of the home agents (simulated and expected)

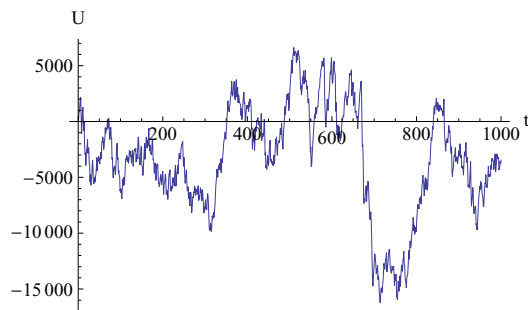


Fig. 4. Cumulated utility of the aggregator with 100 agents as time progresses

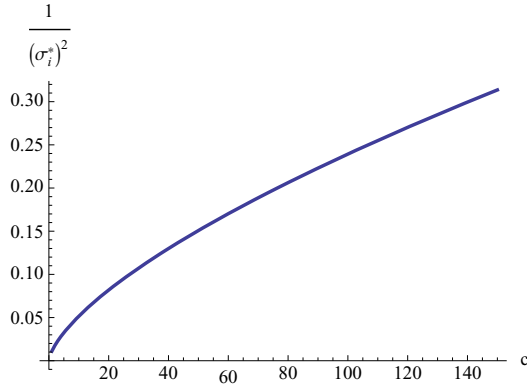


Fig. 5. Required forecast precision as cost parameters increase

with more home agents can achieve better performance, even when the forecasts are less precise.

Fig. 4 shows the simulated cumulative utility of the aggregator agent. Although its expected value is zero as it was proved, its realised value can be arbitrary, oscillating around zero.

In a smart grid the cost parameters can change from hour to hour, therefore it is important to study the behaviour of the system under different costs. In the next experiment we fix $n = 100$ and the ratios $b/c = 1.7$ and $c/s = 2$, therefore the cost parameters change proportionately. Fig. 5 illustrates that as the purchasing costs increase, it becomes more important to possess more precise forecasts. In addition, it can be noted that if the ratio between the cost parameters is fixed, the K_{cbs} value is linear in c (see (6)), and thus γ too.

6 Conclusions and Future Works

This paper investigated a mechanism design problem in a setting where several agents can generate estimates of independent future demands at a cost. An aggregator agent elicits the forecasts and based on this information, optimises a procurement decision. A novel scoring rule based mechanism has been developed which is incentive compatible, efficient, individually rational and ex ante budget balanced, contrary to prior works in this field. Furthermore, the proposed mechanism respects privacy, since forecasts and sensitive consumption data have to be known only by the affected home agent and the aggregator. Several simulation runs confirmed that even a small group of agents can significantly increase their utility by forming such a mechanism compared to purchasing directly from the market. Further on, a relatively small group can achieve almost as much benefit as a larger one.

A practical extension of the presented model is when each agent has different α_i values. In this case calculating the optimal forecast precisions leads to a

non-linear optimisation problem with n variables. Consequently, it may be necessary to apply different γ_i parameters for the home agents, and for this purpose, the α_i parameters have to be common knowledge. From (18) with an arbitrary γ these parameters can be calculated as $\alpha_i = \gamma(\sigma_i^*)^3$, thus cost information can be elicited after an initial step. However, the above derivation is only a sketch and needs further analysis.

Acknowledgments. This work has been supported by the KMR_12-1-2012-0031 (E+Grid) grant and by the János Bolyai scholarship No. BO/00659/11/6.

References

1. Katz, R.H., et al.: An information-centric energy infrastructure: The Berkeley view. *Sustainable Computing: Informatics and Systems* 1(1), 7–22 (2011)
2. Egri, P., Váncza, J.: A distributed coordination mechanism for supply networks with asymmetric information. *Eur. J. of Op. Res.* 226(3), 452–460 (2013)
3. Apt, K.: A primer on strategic games. In: Apt, K.R., Graedel, E. (eds.) *Lectures in Game Theory for Computer Scientists*, pp. 1–37. Cambridge University Press (2011)
4. Zohar, A., Rosenschein, J.S.: Mechanisms for information elicitation. *Artificial Intelligence* 172(16-17), 1917–1939 (2008)
5. Chen, Y., Pennock, D.M.: Designing markets for prediction. *AI Magazine* 31(4), 42–52 (2010)
6. Papakonstantinou, A., Rogers, A., Gerding, E., Jennings, N.: Mechanism design for the truthful elicitation of costly probabilistic estimates in distributed information systems. *Artificial Intelligence* 175(2), 648–672 (2011)
7. Blumsack, S., Fernandez, A.: Ready or not, here comes the smart grid! *Energy* 37(1), 61–68 (2012)
8. Jackson, J.: Improving energy efficiency and smart grid program analysis with agent-based end-use forecasting models. *Energy Policy* 38(7), 3771–3780 (2010)
9. Rose, H., Rogers, A., Gerding, E.H.: A scoring rule-based mechanism for aggregate demand prediction in the smart grid. In: *11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, pp. 661–668 (2012)
10. Egri, P., Váncza, J.: Supply network coordination by vendor managed inventory – a mechanism design approach. In: *Proc. of the 2nd Workshop on Artificial Intelligence and Logistics (AILog 2011)*, *22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 19–24 (2011)
11. Qin, Y., Wang, R., Vakharia, A.J., Chen, Y., Seref, M.M.: The newsvendor problem: Review and directions for future research. *European Journal of Operational Research* 213(2), 361–374 (2011)
12. Ketzenberg, M.E., Rosenzweig, E.D., Maruchek, A.E., Metters, R.D.: A framework for the value of information in inventory replenishment. *European Journal of Operational Research* 182(3), 1230–1250 (2007)
13. Oliveira, F.S., Ruiz, C., Conejo, A.J.: Contract design and supply chain coordination in the electricity industry. *Eur. J. of Op. Res.* 227(3), 527–537 (2013)
14. Chalkiadakis, G., Robu, V., Kota, R., Rogers, A., Jennings, N.: Cooperatives of distributed energy resources for efficient virtual power plants. In: *10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, pp. 787–794 (2011)

Translating Statecharts-Based into BDI Agents: The DSC/PROFETA Case

Giancarlo Fortino¹, Wilma Russo¹, and Corrado Santoro²

¹ Università della Calabria
DIMES

Via P. Bucci, cubo 41c - 87036 - Rende (CS) - Italy
{g.fortino,w.russo}@unical.it

² Università di Catania

Dipartimento di Matematica e Informatica
Viale Andrea Doria, 6 - 95125 - CATANIA - ITALY
santoro@dmi.unict.it

Abstract. In the agent-oriented computing area, different programming models have been to date proposed to design and implement multi-agent systems (MAS). Among them, the Belief-Desire-Intention (BDI) and state machine-based models have emerged as they effectively support the definition of rational and protocol-oriented agent behaviors, respectively. In this paper, we investigate the translation between such models by using two available agent modeling languages: PROFETA, which is based on the BDI agent model, and the agent-oriented Distilled StateCharts (DSC) formalism, which allows modeling agent behaviors based on Statecharts-like state machines. In particular, we propose a *mapping* from DSC to PROFETA that can enable, from one hand, visual programming of PROFETA programs to facilitate the development of PROFETA-based MAS and, from the other hand, simulation of DSC-based agent systems through the execution support offered by the PROFETA engine.

1 Introduction

Agent-oriented computing is a well-established high-level paradigm for the modeling and implementation of intelligent distributed systems in terms of *multi-agent systems* (MAS) [1]. Differently from object-oriented programming, where a unique reference object model exists, many different agent programming models have been to date proposed. Among them, two main models for the definition of agent behaviors emerged respectively from the Artificial Intelligence and the Distributed Systems communities: Belief-Desire-Intention (BDI) [2] and State Machine-oriented [3].

The BDI model [2], which is based on a theory of human practical reasoning [4], assumes that the mental state of an agent consists of three fundamental attitudes: *Beliefs*, corresponding to the agent's knowledge, *Desires*, i.e. the motivational state of an agent and thus its goals, and *Intentions*, which are the

actions to perform to reach the goal(s) and thus are related to deliberative state of an agent. AgentSpeak(L) [5], together with its implementation JASON [6], is one of the best known BDI programming language.

As for the State machine-based agent models, they are available in several mainstream agent-oriented programming frameworks (e.g. JADE [7]) and methodologies (e.g. ELDAMeth [8]). Their main characteristics are the appealing graphical features that enable visual programming of MAS, so facilitating rapid prototyping, and a well-defined operational semantics allowing system verification. Moreover, they are very effective in specifying agent interaction protocols.

While these two kind of models use different approaches, they have a common goal: to provide an easy and efficient way to express the behaviour of an autonomous agent. This paper aims at investigating the integration of such models in order to provide synergic and mutual benefits in programming MAS by exploiting both of them. To this purpose, we considered the following implementations: PROFETA [9,10] as BDI-oriented programming language, and the Distilled StateCharts (DSC) formalism [11] as State Machine-based modeling language. PROFETA is a Python implementation of the BDI kernel language AgentSpeak(L); it provides a set of classes to define agent attitudes (beliefs, actions and goals) and, by exploiting operator overloading, it is able to allow the expression of AgentSpeak(L) declarative constructs inside a Python program. On the other hand, the DSC formalism is purposely derived from the Statecharts to visually model the behavior of single-threaded agents through a set of (graphical) constructs modeling reactive and proactive behavior, reducing complexity through hierarchy, and managing history information. The proposal of this paper is therefore centered on a well-formalized mapping from the DSC to PROFETA, an objective which is achieved by introducing (*i*) a formal model for DSCs and (*ii*) a logical framework able to represent DSC entities into PROFETA constructs. This provides a twofold advantage: the implementation of PROFETA programs can be facilitated through visual programming offered by DSC, whereas MAS based on DSC can be straightforwardly simulated through the PROFETA engine.

The rest of this paper is organized as follows. Section 2 introduces the basic concepts on DSC whereas Section 3 provides some basic concepts on PROFETA. In Section 4, the DSC/PROFETA mapping is described in detail. Finally, conclusions summarize the proposal, discussing the mapping benefits and also delineating future work.

2 Overview of Distilled Statecharts

The Event-driven Lightweight Distilled Statecharts-based Agent (ELDA) model [3,8], relies on the following three models: Behavioral, Interaction and Mobility. The *Behavioral* (or computational) model allows the specification of the agent behavior through the definition of the agent states, the transitions among these states, and the agent reactions as atomic action chains attached to these transitions. The *Interaction* model is based on asynchronous events which an agent can

generate and receive. The *Mobility* model allows for the transparent migration of agents and represents a strong mobility model for which the agent mobility grain can be programmed [12]. These models are based on the Distilled StateCharts (DSC) formalism [11], which was purposely derived from Statecharts [13], a visual formalism that gained valuable success in the Software Engineering community thanks to its appealing graphical features: indeed, Statecharts are currently included in the UML as behavioural diagrams for software modeling. DSCs are obtained by means of a distillation process of Statecharts which involves (i) using only the OR-state decomposition (i.e. hierarchy of states), inter-level state transitions, default and history entrances, and (ii) imposing the following constraints on Statecharts:

- Each DSC model must have an enclosing top state.
- States must be empty, i.e., they do not include activity, entry and exit actions. So activity is only carried out under the form of atomic action chains attached to transitions.
- Transitions (apart from default and history entrances) are always labelled by an event.
- Each composite state has an initial pseudostate from which the default entrance originates, which can only be labelled by an action chain.
- Events are implicitly and asynchronously received through an decoupling event queue.
- To explicitly and asynchronously generate events the action language provides the primitive `generate(<event>(<parameters>))`, where `event` is an event instance and `param` is the list of formal parameters of event including the event sender, the event target, and (possibly) a list of specific event parameters. Run-to-completion execution semantics: an event can be processed only if the processing of the previous event has been fully completed.

In the following subsection we first formalize the structure of the DSC state machine (*DSM*) and, then, provide an operational semantics for its execution. In our ELDA model, the agent behavior is specified as a DSM whereas the ELDA execution semantics are established according to the DSM execution semantics.

2.1 DSC Formalization

DSM is formalized by the following tuple:

$$\langle \Sigma, s_0, L, \Phi, defaultHistory, deep, defaultEntrance \rangle$$

where, Σ is the set of states, s_0 is the initial state configuration, L is the set of transition labels, Φ is the set of transitions, *defaultEntrance* and *defaultHistory* indicate the default entrance and the default history entrance respectively and *deep* is an attribute of an history pseudostate indicating deep or shallow history.

Σ is the set of the following disjoint sets: Σ_{cs} (composite states), Σ_{ss} (simple states), $\Sigma_{fs} = fs$ (final state), and Σ_H (history pseudostates). In particular, initial pseudostates are not explicitly named since each composite state has an initial pseudostate, whereas the history pseudostates are denoted by $hs_i \forall i = 1.. | \Sigma_H |$.

To provide a representation for state configurations which explicitly encode the state hierarchy, state configurations are represented as terms over the signature F . In particular:

- The set of state configurations is the set of (first-order) terms $T(F, X)$ ¹ over F and X , where X is a denumerable set of variable symbols and $F = F_0 \cup F_1$, $F_0 = \Sigma_{ss} \cup \Sigma_{fs} \cup \Sigma_H$ and $F_1 = \Sigma_{cs}$ which implies that $F = \Sigma$.
- A generic state configuration of DSM is a monadic term, i.e., it is a word spelled with unary symbols (from Σ_{cs}) and ending in a constant (from $\Sigma_{ss} \cup \Sigma_{fs} \cup \Sigma_H$) or a variable (from X).
- The current state configuration (CSC) of a DSM is a term of $T(\Sigma, \emptyset)$ where the history subterm (or history pseudostate), if any, has been replaced by its current sub-configuration by means of the function $history : T(\Sigma_H, \emptyset) \rightarrow T(\Sigma - \Sigma_H, \emptyset)$.

A transition is a triple $\langle src, tgt, l \rangle$ where $src \in T(\Sigma, X)$, $tgt \in T(\Sigma, \emptyset)$, $l \in L : l = e[g]/ac$ and $e \in E$ (the set of triggering events), $ac \in L_{ac}$ (the language of the actions), $g \in L_g \subset L_{ac}$ (the language of the guards). In particular:

- A transition is a term-rewriting rule in the sense that the pair $\langle src, tgt \rangle$ is the actual rewriting rule and the associated l embodies the pre-condition by which the rule can be applied.
- The set of transitions or transition relation, $\Phi \subseteq T(\Sigma, X) \times T(\Sigma, \emptyset) \times L$, represents the term-rewriting system.

In Figure 1 the diagram of an example of a DSC is reported along with its representation based on the above formalized rewriting system.

The DSC execution semantics are defined in terms of an abstract machine, which embodies a DSM , whose key components are:

- An event queue (EQ), which holds incoming event instances until they are dispatched;
- An event dispatching mechanism (EDM), which selects and de-queues event instances from the event queue.
- An event processor (EP), which processes dispatched events.

More concretely the abstract machine is represented by an ELDA agent itself whose architecture, beside the EQ , includes the EDM and EP components supported by the single thread of control. The semantics of event processing is based on the *run-to-completion* (RTC) assumption which means that an event can only be de-queued and dispatched if the processing of the previous event is fully completed. The processing of a single event by a DSM is known as the *RTC step*, which implies that, before commencing on an RTC step, a DSM

¹ In general, given a disjoint set $F = \bigcup_{n \geq 0} F_n$ of function symbols called a signature and a (denumerable) set X of variable symbols, the set of (first-order) terms $T(F, X)$ over F and X is the smallest set containing X such that $f(t_1, \dots, t_n)$ is in $T(F, X)$ whenever $f \in F_n$ and $t_i \in T(F, X) \forall i = 1..n$. In a well-formed term, each function symbol of arity n has n immediate subterms.

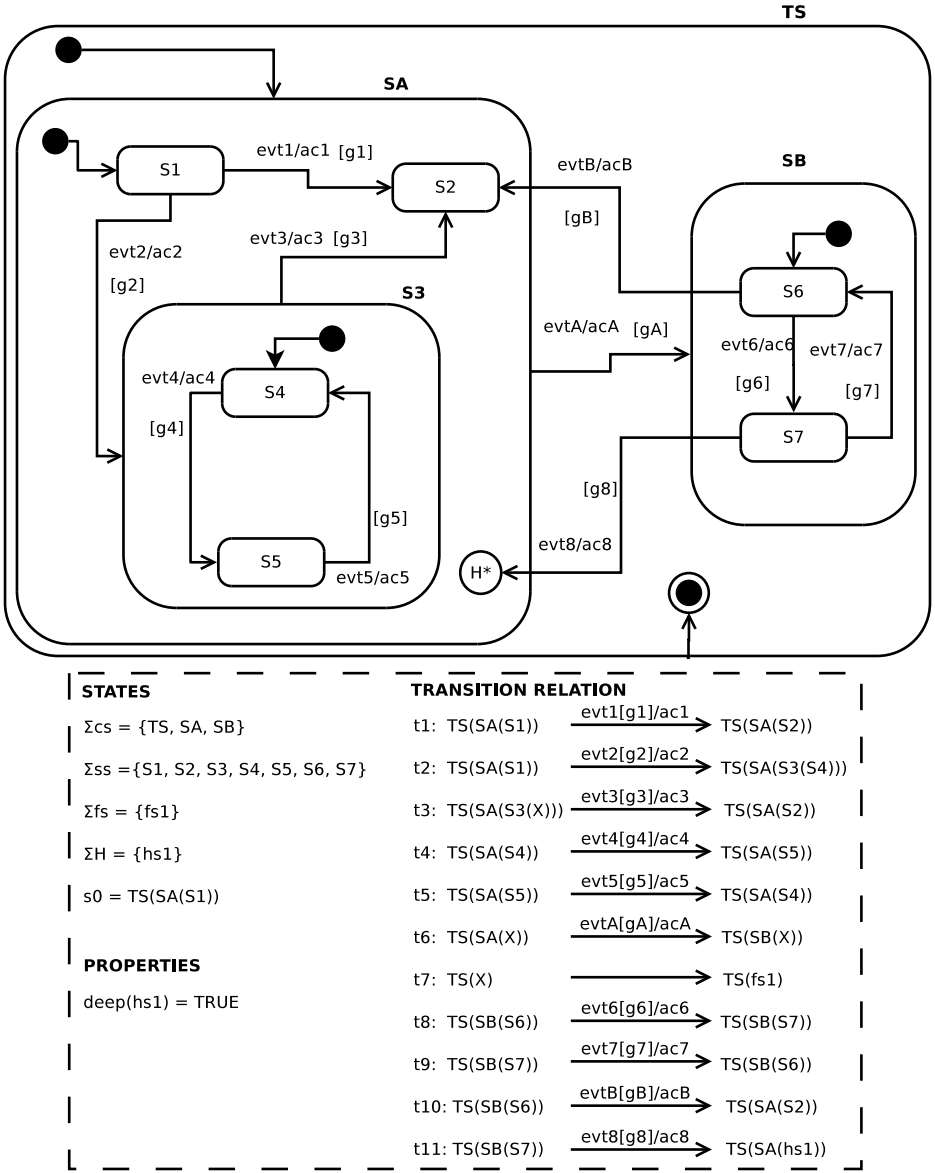


Fig. 1. An example of a DSC with its corresponding term-rewriting-based version

is in a stable state configuration with the action expression completed and no events can be processed while a *DSM* is in some intermediate and inconsistent situation. Given the lastly de-queued event by *EDM*, the main task of *EP* is to cyclically:

1. Find out which transitions are enabled, i.e., which transitions could be fired based on the de-queued event;
2. Select, among the enabled transitions, the transitions to be fired. Only one transition of a *DSM* is step-by-step ready to be fired.

3 Overview of PROFETA

PROFETA (*Python RObotic Framework for dEsigning sTrAtegies*) [9,10] is a Python framework for programming autonomous systems, like agents or robots, using a declarative approach. It provides a declarative language which is a dialect of AgentSpeak(L) [5], a well-known BDI kernel language [2,4]. PROFETA exploits some peculiar features of Python, mainly object-orientation and operator overloading, in order to add declarative constructs to Python programs, while a proper set of classes provides the suitable abstractions to support the specific concepts introduced by the BDI model.

3.1 PROFETA Entities

In order to implement BDI agents, PROFETA provides the following basic entities: *beliefs*, *actions* and *goals*.

Beliefs are expressed using logic atomic formulae with ground terms, which can be any valid Python type with the only exception that strings starting in uppercase represent *variables*. Syntax is Prolog-like, therefore expressions like `my_position(1230,450)`, `obstacle_position("in_front")` or `object_got()`, are valid beliefs. On the other hand, the expression `my_position("X","Y")` binds the free variables X and Y, respectively to the first and second argument of the belief `my_position`. Beliefs play the same role of *facts* in an expert system; they can be *asserted*, if they represent a true situation², or *retracted* when they no longer hold. A PROFETA program has its own *belief base (BB)* acting as a repository of beliefs asserted during program execution.

PROFETA provides also special beliefs called *reactors* which are used to represent *one-shot events*: while the life time of a belief (i.e. when it is asserted and/or retracted) is completely under the control of the programmer, a reactor is automatically removed from the BB once it has been used for the first time. Its role, in a PROFETA program, will become more clear in Section 3.3.

In a PROFETA program, beliefs and reactors must be declared by subclassing respectively `profeta.attitude.Belief` and `profeta.attitude.Reactor` framework classes.

² A situation which the agent believes as true.

Actions represent computations that are executed *atomically* to allow the agent to “do something”. Examples of actions are: activating an arm to pick an object, making a robot to go to a certain position, sampling an image through the camera, etc. From the syntactic point of view, an action is represented with an atomic formula with zero or more parameters; a parameter can be a constant or a bound variable. Therefore, expressions like `move_to(1500,1000)` or `activate_arm("X")`, where "X" is a bound variable, are valid action representations. Also actions must be declared before being used; this is performed by defining a sub-class of `profeta.attitude.Action` and implementing the action code (in Python) in the `execute()` method.

Goals represent states in which a certain specific objective has been fulfilled by the agent. Such an objective can be reached by performing a certain—more or less complex—sequence of actions; from the syntactical point of view, goals are represented as the other basic entities, i.e. by means of atomic formulae with zero or more parameters. A goal is defined as a sub-class of `profeta.attitude.Goal`.

3.2 PROFETA Program Structure and Syntax

A PROFETA is basically composed of two parts. The first part contains the declaration of all the objects used in the program, i.e. the specific beliefs, reactors, actions and goals. The second part is the program body which is based on the expression and execution of a set of *reactive rules*; each rule is triggered by the occurrence of a certain event and executes a specific computation in order to handle the said event. This model captures the nature of a agent’s behaviour which is intrinsically reactive: indeed an agent cyclically senses the environment and, on this basis, if sensors detect a specific state or measure, or the occurrence of a certain event, it performs proper actions in response to such a sense.

Each PROFETA rule has (i) an *header*, expressing the *event*; (ii) a *condition* which must be true for the rule to be triggered; and (iii) a *body*, which represents the (set of) action(s) to be performed when the rule is triggered. This is expressed by means of the following syntax:

event “/” *condition* “>>” “[” *body*, i.e. list of actions “]”

Events which can trigger a rule are (syntax is reported in braces):

1. The assertion of a given belief (+*belief*) or reactor (+*reactor*).
2. The retraction of a given belief (−*belief*).
3. The request to achieve a certain goal (~*goal*).

Conditions refer to the state of the belief base, that is, they express which beliefs (and with which parameters) must be asserted in the BB in order to allow the execution of a triggered rule. Expressing the condition is optional.

The *body* of a rule contains the things to be done in executing a triggered rule; it can include:

1. The execution of *atomic actions*.
2. The assertion or retraction of a belief or a reactor (+*belief* or +*reactor*).
3. The request to achieve a specific goal (~*goal*).

```

1 class object_position(Belief): pass
2 class obstacle_on_the_way(Reactor): pass
3 class go(Goal): pass
4 class detect_object(Goal): pass
5 class reach_object(Goal): pass
6 class pick_object(Action): def execute(self): ....
7 class scan_scene(Action): def execute(self): ....
8 class move_to(Action): def execute(self): ....
9 class avoid_obstacle(Action): def execute(self): ....
10
11 ~go() >> [ ~detect_object(), ~reach_object(), pick_object(),
12           -object_position("_", "_"), ~go() ]
13 ~detect_object() / not(object_position("_", "_")) >> [ scan_scene(),
14                                                       ~detect_object() ]
15 ~detect_object() / object_position("X", "Y") >> [ ]
16 ~reach_object() / object_position("X", "Y") >> [ move_to("X", "Y") ]
17 +obstacle_on_the_way() >> [ avoid_obstacle(), ~reach_object() ]

```

Fig. 2. An Example of a PROFETA Program

3.3 A Simple PROFETA Program

In order to let the reader to understand PROFETA syntax, we will provide here an example of a PROFETA program which includes all the constructs of the language. Interested readers may find further details in [10,9]. The example will also show the difference between beliefs and reactors.

Let us consider a mobile robot whose aim is to detect objects in the environment and to pick them; to this aim, the robot is equipped with an artificial vision system and a mechanical arm. The program is sketched in Figure 2.

The main goal is `go()` (line 11), whose body reports the request to first achieve the goal `detect_object()`, then the goal `reach_object()` and then to execute the (atomic) action `pick_object()`, which implies to drive the arm to perform object picking; the last element of the body rule is the request to achieve again the goal `go()`: this way, by means of recursion, a continuous loop can be implemented.

Things done by the goal `detect_object()` are based on the presence of the belief `object_position(X,Y)`, which represents the location of the object to be picked. If such a belief is not asserted (line 13), then `scan_scene()` action is executed and then the goal is called recursively. The said action has the responsibility of activating the vision system and, if it has identified an object, properly asserting the belief `object_position(X,Y)`. In the latter case, when the goal is recursively called, the rule in line 15 is executed (no action here, because the goal has been achieved).

The third goal implies to execute the action `move_to(X,Y)`, which drives the robot towards the position in which the object has been located. In performing such an action, if an obstacle is met in the path, we suppose that the `obstacle_on_the_way()` reactor is automatically asserted, thus causing triggering of the rule in line 18; as a consequence, action `avoid_obstacle()` is first executed and then goal `reach_object()` is triggered again. This last rule can be used to explain the semantics of a reactor with respect to a belief: indeed,

a reactor is a special kind of belief that, once its assertion causes the triggering of a rule, it is automatically removed from the BB; reactors can thus be used to represent a condition in which an event occurred: once the event has been consumed, the condition no more holds and thus its representation can be deleted.

4 Mapping DSCs onto PROFETA Programs

The background on DSC and PROFETA, provided in the previous Sections, shows that both models feature some similarities. The DSC concepts of *states*, *guards* and *events* are well captured, in PROFETA, by *beliefs*, *conditions* and *reactors*, while the concept of *transition* can be easily expressed by means of a PROFETA rule. Despite these similarities, there are some important differences which make DSC implementation not so straightforward: OR decomposition and shallow and deep histories are constructs strongly based on the concept of *hierarchy of states* and, since PROFETA programs are *flat*, implementing such constructs requires a proper modeling process.

4.1 Representing Hierarchies and States

Let us consider, as a first hypothesis, that state names are *unique* in the whole DSC. In order to model macrostates and hierarchies, let us introduce the PROFETA belief `parent`(*SP*, *S*) which represents the parent relationship “state *SP* is a *parent* of *S*”. With reference to the DSC in Figure 1, in order to represent the hierarchy, the following beliefs must be initially asserted:

$$\begin{aligned} & \text{parent}(\text{ts}, \text{sa}), \text{parent}(\text{ts}, \text{sb}), \\ & \text{parent}(\text{sa}, \text{s1}), \text{parent}(\text{sa}, \text{s2}), \text{parent}(\text{sa}, \text{s3}), \\ & \text{parent}(\text{s3}, \text{s4}), \text{parent}(\text{s3}, \text{s5}) \end{aligned}$$

According to the DSC semantics, each state has either only one parent or no parents, therefore the following first-order logic formula holds:

$$\forall s, \exists^1 s' \neq s : \text{parent}(s', s) \vee \nexists s'' : \text{parent}(s'', s) \quad (1)$$

Belief `parent`() is used to represent the hierarchical structure of a DSC; it is a *static* information derived at design time. Indeed, we can suppose that the resulting PROFETA program, before starting, would have such beliefs properly asserted in the Belief Base (BB).

On the other hand, at runtime, we must use something which is able to represent the evolution of the DSC itself by capturing its current state. We model this aspect, that is, the fact that the DSC, at a certain time instant, is in a state \mathbf{s}^* , by means of the belief `current`(\mathbf{s}^*). Obviously, a single instance of such belief is allowed: when the current state changes from s' to s'' , belief `current`(s') will have to be replaced by `current`(s''), i.e. :

$$\exists^1 s : \text{current}(s) \quad (2)$$

According to the OR decomposition model, when a DSC is in a state s^* , it is also in *all parent states* of s^* . For example, with reference to Figure 1, when we are in state $s2$, we are *also* in states sa and ts ; similarly, being in state $s5$ means to be also in states $s3$, sa and ts . To model OR decomposition, we introduce the belief $in()$ with the following meaning: given that a DSC is in a certain state s^* , supposing that such a state is included in states s_1, s_2, \dots, s_k in a OR-decomposition relationship, then the following set of beliefs will be asserted $\{in(s^*), in(s_1), in(s_2), \dots, in(s_k)\}$. The $in()$ belief is thus strictly related to $parent()$: if the belief $in(s)$ is asserted and s has a parent, i.e. belief $parent(s', s)$ is asserted, then also belief $in(s')$ must be asserted. This condition can be represented with the following implication:

$$\forall s, in(s) \wedge \exists s' : parent(s', s) \Rightarrow in(s') \quad (3)$$

For each state s of the DSC, we can thus build the set $Bel^{(s)}$ which contains all the $in()$ beliefs obtained by first asserting $in(s)$ and then applying implication (3) recursively.

Therefore, with reference to Figure 1, we can derive the following $Bel^{(s)}$ sets:

$$\begin{aligned} Bel^{(ts)} &:= \{in(ts)\} \\ Bel^{(sa)} &:= \{in(sa), in(ts)\} \\ Bel^{(s1)} &:= \{in(s1), in(sa), in(ts)\} \\ Bel^{(s2)} &:= \{in(s2), in(sa), in(ts)\} \\ Bel^{(s3)} &:= \{in(s3), in(sa), in(ts)\} \\ Bel^{(s4)} &:= \{in(s4), in(s3), in(sa), in(ts)\} \\ Bel^{(s5)} &:= \{in(s5), in(s3), in(sa), in(ts)\} \\ Bel^{(sb)} &:= \{in(sb), in(ts)\} \\ Bel^{(s6)} &:= \{in(s6), in(sb), in(ts)\} \\ Bel^{(s7)} &:= \{in(s7), in(sb), in(ts)\} \end{aligned}$$

We can now express two other mandatory conditions which must both hold when the DSC is in a state s :

1. All the beliefs of the set $Bel^{(s)}$ must be asserted;
2. No belief $in(s')$ not belonging to $Bel^{(s)}$ can be asserted.

These conditions can be expressed as:

$$\forall s, in(s) \in BB \wedge in(s) \in Bel^{(s)} \quad (4)$$

$$\forall s, \nexists s' : in(s') \in BB \wedge in(s') \notin Bel^{(s)} \quad (5)$$

4.2 Representing Transitions

A DSC transition is represented by an arrow exiting from a *starting state* “ ss ” and entering into an *ending state* “ es ”; it is activated following a certain *event*

“*evt*” given that a specified *guard* “*g*” is true; its activation also triggers the execution of a certain *action* “*act*”.

When a DSC is in a given state s^* , candidate transitions for activation are only those whose starting state ss is s^* itself or one of the states in the parent chain of s^* , if s^* is included in a OR-decomposition hierarchy. Having introduced the set $Bel^{(s)}$ above, we can state that *candidate transitions are only those whose starting state “ ss ” is such that $in(ss) \in Bel^{(s^*)}$.*

On this basis, the head of the PROFETA rule representing a transition can be simply written as:

```
+evt() / (in(ss) & g) >> [ .... ]
```

The consequence of the activation of a transition is the execution of an action and the change of state, from ss to es . The latter operation (state change) requires a deeper analysis, since *entering into the new state es* could involve other transitions, on the basis of the structure of the DSC. The problem is once again related to macro-states, since entering in one of such states implies to follow the *start transition* of the inner statechart. With reference to Figure 1, entering into sa implies to reach (internal) state $s1$, entering into $s3$ implies to reach the substate $s4$, and entering into sb implies to reach $s6$.

The process of changing the state from ss to es can be thus described as composed of the following operations:

1. First the state ss is left; to this aim, the current state cs is determined, belief **current**(cs) is removed, and the same happens for all beliefs included in set $Bel^{(cs)}$; we consider the presence of an ad-hoc library action, called **leave_current**(), which performs the said operations. This action executes also other activities which are related to shallow and deep histories (see Section 4.3).
2. An internal event is generated to signal that we are entering into a new state; this is performed by means of reactor **enter**(es).
3. A specific rule is present to respond to event **+enter**(es), the action depends on the nature of es , and it is detailed in the items below;
4. If es is a simple state, it is immediately reached by asserting **current**(es) and all the beliefs in $Bel^{(es)}$; these assertions can be performed by another ad-hoc library action, called **state_to**(es).
5. If es is a macro-state, the starting transition is followed and a new **+enter**(new_es) event is generated accordingly.

We are now ready to represent, in a PROFETA program, all transitions in the DSC of Figure 1, with the only exception of the deep history symbol which will be instead dealt with in the next Subsection; the relevant listing is thus reported in Figure 3.

4.3 Representing Histories

Representing DSC histories in a PROFETA program implies to perform certain tasks in order (*i*) to *save the state left*, when we exit from a state or macrostate

```

1  +start() >> [ +enter("ts") ]
2  +enter("ts") >> [ +enter("sa") ]
3
4  # macro-state SA
5  +enter("sa") >> [ +enter("s1") ]
6  +evtA() / (in("sa") & gA) >> [ acA(), leave_current(), +enter("sb") ]
7
8  # macro-state S1
9  +enter("s1") >> [ state_to("s1") ]
10 +enter("s2") >> [ state_to("s2") ]
11 +enter("s3") >> [ +enter("s4") ]
12 +evt1() / (in("s1") & g1) >> [ ac1(), leave_current(), +enter("s2") ]
13 +evt2() / (in("s1") & g2) >> [ ac2(), leave_current(), +enter("s3") ]
14
15 # macro-state S3
16 +enter("s4") >> [ state_to("s4") ]
17 +enter("s5") >> [ state_to("s5") ]
18 +evt3() / (in("s3") & g3) >> [ ac3(), leave_current(), +enter("s2") ]
19 +evt4() / (in("s4") & g4) >> [ ac4(), leave_current(), +enter("s5") ]
20 +evt5() / (in("s5") & g5) >> [ ac5(), leave_current(), +enter("s4") ]
21
22 # macro-state SB
23 +enter("sb") >> [ +enter("s6") ]
24 +enter("s6") >> [ state_to("s6") ]
25 +enter("s7") >> [ state_to("s7") ]
26 +evt6() / (in("s6") & g6) >> [ ac6(), leave_current(), +enter("s7") ]
27 +evt7() / (in("s7") & g7) >> [ ac7(), leave_current(), +enter("s6") ]
28 +evtB() / (in("s6") & gB) >> [ acB(), leave_current(), +enter("s2") ]
    
```

Fig. 3. Listing of the PROFETA Program related to the DSC in Figure 1

which includes a **H** or **H*** symbol, and *(ii)* to *restore it* when we follow a transition leading to that history symbol. The information to save and restore depends on the *type* of the history, i.e. shallow (**H**) or deep (**H***).

In order to represent the history symbol in the structure of the DSC, we introduce the beliefs `hist(s)` and `deep_hist(s)` to indicate that state s includes respectively a shallow and deep history symbol. Similarly to `parent()`, these beliefs have to be initially asserted to represent the structure of the DSC. Obviously, for each state s either `hist(s)` or `deep_hist(s)` can be asserted, or nothing. By exploiting the said beliefs, we can understand if we need to save something when we leave a state through a certain transition.

As stated in Section 4.2, leaving a state concretely implies to execute the action `leave_current()` which clears the `in()` beliefs of the set $Bel^{(cs)}$, where cs is the current state. However, in such a clearing operation, the said action has to perform proper checks on history symbols and, on this basis, determine the history information to be saved, as detailed in the following.

Shallow Histories. Let us first consider shallow history. If we are leaving current state cs , if s such as $in(s) \in Bel^{(cs)}$ has a history symbol, the *internal state* of s , say si , must be saved. Finding si is quite simple since it is such to verify the following condition:

$$hist(s) \wedge in(s) \in Bel^{(cs)} \wedge parent(s, si) \wedge in(si) \in Bel^{(cs)} \quad (6)$$

Once the state si has been found, to save it we have to properly assert the belief $\text{saved}(si, hs)$, where si is the saved state and hs is the state the history symbol belongs to.

Deep Histories. In order to represent deep histories, we have to change the way in which the state to be saved is determined. To this aim, in leaving state cs , if s such as $in(s) \in Bel^{(cs)}$ has a deep history symbol, we have to determine the *deepest internal state* and save it. However, given the way in which the set $Bel^{(cs)}$ is built, such a deepest state is always cs since it is the sole state in the set which has *no children*. On this basis, operations to be performed on the occurrence of a deep history symbol are more simple than those of shallow history and can be summarised by the following logic formula:

$$\exists s, in(s) \in Bel^{(cs)} \wedge deep_hist(s) \Rightarrow saved(cs, s) \quad (7)$$

Entering into Histories. Once we have determined the operations to perform to save the state left in the presence of a history symbol hs , entering into such a history simply implies to restore the saved state: if the belief $\text{saved}(si, hs)$ is asserted, then si will be the state to reach. To represent the event which triggers the “*entering into a history*” transition, we use the reactor $\text{enter}(\text{"hist"}, \text{"hs"})$, therefore the complete operation can be represented with the following piece of code:

```
+enter("hist","hs") / saved("X", "hs") >> [ +enter("X") ]
```

On this basis, with reference to Figure 1, the complete transition related to event $evt8$ can be written as follows:

```
1 +evt8() / (in("s7") & g8) >> [ ac8(), leave_current(), +enter("hist","sa") ]
2 +enter("hist", "sa") / saved("X", "sa") >> [ state_to("X") ]
```

Adding these two lines of code to the listing in Figure 3 completes the PROFETA program related to the whole statechart in Figure 1.

5 Conclusions

This paper has dealt with the integration of two agent programming tools based on different agent models: DSC, a visual tool for agent behaviour specification through the statechart formalism, and PROFETA, a BDI agent programming language. Through a logical framework, aiming at ensuring the preservation of DSC semantics, and by using a reference example, a proper mapping of DSC constructs into PROFETA entities and statements has been derived.

As future work, we plan to include the derived DSC/PROFETA translation rules into ELDATool [8]—a visual software tool designed to engineer MAS through DSC—with the objective of allowing the automatic generation of the PROFETA code from DSC-based behaviours. This is able to enrich the features of ELDATool with the possibility of performing a rapid prototyping of PROFETA agents.

References

1. Weiss, G. (ed.): *Multiagent Systems*. The MIT Press (April 1999)
2. Rao, A., Georgeff, M.: BDI agents: From theory to practice. In: *Proceedings of the First International Conference on Multi-agent Systems (ICMAS 1995)*, San Francisco, CA, pp. 312–319 (1995)
3. Fortino, G., Garro, A., Mascillaro, S., Russo, W.: Using event-driven lightweight dsc-based agents for mas modelling. *IJAOSE* 4(2), 113–140 (2010)
4. Bratman, M.E.: *Intentions, Plans and Practical Reason*. Harvard University Press (1987)
5. Rao, A.: *AgentSpeak (L): BDI agents speak out in a logical computable language*. In: Perram, J., Van de Velde, W. (eds.) *MAAMAW 1996*. LNCS, vol. 1038, pp. 42–55. Springer, Heidelberg (1996)
6. Jason Home Page (2004), <http://www.jason.sourceforge.net/>
7. Fortino, G., Rango, F., Russo, W.: Engineering multi-agent systems through statecharts-based jade agents and tools. *T. Computational Collective Intelligence* 7, 61–81 (2012)
8. Fortino, G., Russo, W.: Eldameth: An agent-oriented methodology for simulation-based prototyping of distributed agent systems. *Information & Software Technology* 54(6), 608–624 (2012)
9. Fichera, L., Marletta, D., Santoro, C., Nicosia, V.: A Methodology to Extend Imperative Languages with AgentSpeak Declarative Constructs. In: *Workshop on Objects and Agents (WOA 2010)*, Rimini, Italy, September 5-7. CEUR-WS Publisher (2010) ISSN 1613-0073
10. Fichera, L., Marletta, D., Nicosia, V., Santoro, C.: Flexible Robot Strategy Design Using Belief-Desire-Intention Model. In: *Obdržálek, D., Gottscheber, A. (eds.) EU-ROBOT 2010*. CCIS, vol. 156, pp. 57–71. Springer, Heidelberg (2011)
11. Fortino, G., Russo, W., Zimeo, E.: A statecharts-based software development process for mobile agents. *Information & Software Technology* 46(13), 907–921 (2004)
12. Fortino, G., Rango, F.: An application-level technique based on recursive hierarchical state machines for agent execution state capture. *Sci. Comput. Program.* 78(6), 725–746 (2013)
13. Harel, D.: Statecharts: A visual formalism for complex systems. *Sci. Comput. Program.* 8(3), 231–274 (1987)

Exploiting MAS-Based Simulation to Improve the Indian Railways' Efficiency

Supriyo Ghosh¹, Animesh Dutta¹, Viviana Mascardi², and Daniela Briola²

¹ Department of Information Technology
National Institute of Technology
Durgapur 713209, India

ghosh.supriyo.cse@gmail.com, animesh.dutta@it.nitdgp.ac.in

² Department of Informatics, Bioengineering, Robotics and System Engineering
University of Genova

via Dodecaneso 35, 16146, Genova, Italy

{viviana.mascardi,daniela.briola}@unige.it

Abstract. Despite being one of the world's largest railways networks, with a daily transportation of over 25 million passengers and 2.8 million tons of freight, the Indian Railways perform their signaling, traffic management and trains scheduling activities in a completely manual way.

The lack of automation causes not only significant delays, but also frequent collisions where passengers die or remain seriously injured.

This paper describes how we modeled the Indian Railways system as a MAS and the results of the simulations run using NetLogo on real data retrieved from the Indian Railways dataset. The simulated system is – under some assumptions – collision-free and, thanks to the integration of the MAX-SUM algorithm, minimizes the individual train delay.

1 Introduction

Indian Railways (IR) is one of the world's largest railways networks comprising 115,000 km of track over a route of 65,000 km and 7,500 stations. In 2012, it transported over 25 million passengers daily (over 9 billion on an annual basis). In 2011, the IR carried over 8,900 million passengers annually and more than 24 million passengers daily (roughly half of which were suburban passengers) and 2.8 million tons of freight daily.

In spite of this huge traffic of people and goods, signaling, traffic management and trains scheduling are carried out by the IR staff who communicates and coordinates in a completely manual way. This explains the need of 1.4 million of employees¹ and the impressive amount of collisions that take place², almost always causing deaths and serious injuries.

The problems of the IR manually controlled railway system include:

¹ Data referred to 2011: http://www.indianrailways.gov.in/railwayboard/uploads/directorate/stat_econ/Stat_0910/Year%20Book%202009-10-Sml_size_English.pdf, accessed on May, 2013.

² http://en.wikipedia.org/wiki/List_of_Indian_rail_incidents, accessed on May, 2013.

- The control room of a station can control a train only if it is within the range of 365 meters, which is an almost limited range.
- The train driver has to look out the train for the signal board even while driving during night and in foggy or bad weather.
- Computing the exact distance of the train from a signal post is difficult, and controlling the train's speed in order to ensure that it will stop in time if the next signal will require so, is also hard.
- The signaling system is fixed and signals cannot convey information of speed restrictions; such restrictions are communicated from the crew operating on a train who realizes the need of a speed restriction, to the control room in the next station, by writing a report when they reach the station.
- Tracks management is performed by means of high frequency walkie-talkie communication between the control room staff and the train driver or crew members; being an entirely human-based process, this activity is subject to many errors.

Besides these problems, trains are scheduled manually and the resulting timetable does not assume that all trains operate on full speed, which decreases the throughput of the system in terms of resource utilization; also, the timetable includes redundant waiting time for some trains, creating unnecessary delays.

A specification stating the functional requirements of a Centralized Traffic Control (CTC)/Traffic Management System (TMS) has been recently released by the Ministry of Railways³. The CTC/TMS should carry out the following two categories of functions:

1. centralized operations of signaling for a large area encompassing multiple interlocked stations;
2. centralized real time monitoring of trains traffic for enabling efficient decisions making for traffic control of large areas.

Although the CTC/TMS will definitely improve the current IR situation, a centralized approach does not seem to be the most suitable one for a network where about 10,000 trains run daily and must coordinate among themselves and with the stations' control rooms.

As observed in [2], Agent-Based Models (ABMs) are particularly suited to tackle situations characterized by the presence of a high number of autonomous entities whose behavior (actions and interactions) determines in a non-trivial way the evolution of the overall system. ABMs support the study and analysis of topics like decentralized decisions making, local-global interactions, self-organization, emergence and effects of heterogeneity in the simulated system.

Besides modeling the system, researchers are of course interested in simulating it, hence leading to the Agent-Based Modeling and Simulation (ABMS) approach which is strongly inspired by social simulation and is considered by some scientists as “a third way of doing science” [1].

³ <http://www.rdso.indianrailways.gov.in/works/uploads/File/TMS%20-%20DRAFT%20SPEC.pdf>, accessed on May, 2013.

The motivation for this paper is to move a concrete step towards the development of a fully automated and decentralized IR to overcome collisions due to manual errors and to improve trains punctuality, hence allowing IR to provide a better Quality of Services (QoS) to the customers.

The chosen approach is that of Agent-Based Modeling and Simulation, which perfectly fits the IR domain: in IR we can identify different entities playing different roles, that are autonomous, distributed, concurrent, and interact via asynchronous communication using a high level communication language. Modeling such entities as agents is a very natural choice: simulating the resulting Multiagent System (MAS) to check whether the model is realistic, and to evaluate the effects of interventions in the model itself, can give important hints on how improving the system's efficiency.

In this paper we describe how we modeled the IR system as a MAS and its simulation and results using NetLogo⁴. The IR MAS was already discussed in [12], that describes the negotiation algorithms in detail and the delay optimization based on the MAX-SUM algorithm [16, 17], and in [3], that describes the agents' features in terms of beliefs, desires and intentions.

With respect to the two papers above, the system's model given in this paper is different being based on concepts inspired by dynamic logic (fluent and persistent functions), and the description of the NetLogo implementation and of its results is original. The simulated system is collision-free under specific constraints and, thanks to the integration of the MAX-SUM algorithm, minimizes the individual train delay.

The paper is structured as follow: Section 2 shortly discusses the exploitation of MASs and ABMS techniques in the railway domain; Section 3 describes our new model of the IR system; Section 4 presents six problems with their solution in our MAS; Section 5 reports the results of the experiments executed with NetLogo and lastly Section 6 concludes and describes the future work.

2 Related Work

As observed in a recent survey [8], railway dispatching or scheduling has been usually modeled using classical technologies, such as operational research and constraint programming. These techniques are suitable to model static situations with complete information, but they lack the ability to cope with the dynamics and uncertainty of real railway traffic management. In order to overcome the limitations of traditional centralized and monolithic approaches, many researchers from the multiagent community started working on this domain. Fischer et al. [10] present a MAS that models a society of transportation companies whose goal is to deliver a set of dynamically issued orders satisfying some given cost and time constraints. The proposed conceptual system together with a further study [11] led to two practical applications, i.e., the TeleTruck system [13] and a railroad scheduling system [14]. Cuppari et al. [9] employ a logic

⁴ <http://ccl.northwestern.edu/netlogo/>, accessed on May, 2013.

programming-based environment to prototype a MAS for the management of freight train traffic. The work presented in [4] uses a multiagent approach to the scheduling system for train coupling and sharing. The system is incremental, takes incomplete task specifications into account and generates an initial plan using the contract net protocol: then the post-optimization of the initial solution is achieved by means of the simulated trading protocol.

The agent approach has also been used for other railway applications in the recent years. To model railway access negotiation, Tsang and Ho [18] employ a multiagent approach in which a train services provider (TSP) and an infrastructure provider (IP) are represented by individual software agents. An IP-TSP transaction is simulated by a negotiation protocol. For flexible trains traffic control and conflicts avoidance, Proenca and Oliveira [15] adopt a multiagent railway control system made up of two subsystems: control and learning. The “Control” subsystem is responsible for traffic management and guidance in the network. The “Learning” subsystem has the objective of analyzing the past accumulated situation descriptions and inferring rules that anticipate trains conflicts and improve traffic-control processes. In the set of papers [5–7], Briola et al. describe a multiagent system called FYPA that is used to dynamically manage the allocation of trains on railway tracks inside a station: every train and every railway segment is associated with a dedicated agent, and the real time reallocation of trains is made following a complex negotiation protocol. The system is currently part of the Ansaldo STS (Italy) applications portfolio.

3 Agent-Based Model of the IR System

Model of the IR Physical Network

The IR physical network (*IRN*) can be suitably modeled as a mixed multigraph, namely a graph where some arcs are directed and some are undirected (“mixed graph”), and where multiple arcs with the same source and target vertices are allowed (“multigraph”).

For sake of clarity, we will present *IRN* as if it were a direct multigraph. Undirected arcs impact only in a couple of points of our presentation, where we will deal with them explicitly.

Vertices in V can model either stations (S) or junctions (J), and stations' identifiers are disjoint from junction' ones: $V = S \cup J \wedge S \cap J = \emptyset$. The difference between a station and a junction is that only one train at a time can cross a junction and it cannot stop over it, whereas a station can hold many trains (usually the station's capacity is ≥ 1) and trains can stop on it. Vertices will be identified by v throughout the paper. If needed, we will use subscripts and we will explicitly state whether v_i models a station or a junction by writing s_i and j_k , respectively (since stations and junctions are disjoint, if $v_i = s_i$ represents a station and $v_k = j_k$ represents a junction, then $i \neq k$). Edges E are identified by $e_{(i,j,k)}$, where i is the index of the source vertex v_i , j is the index of the destination vertex v_j , and k ranges on $1, 2, \dots, e$ where e is the total number of

edges directed from v_i to v_j . If there is no need to give details on an edge's source, destination and number, we will use e to represent it.

Agents

Each junction and station in IRN has an agent associated with it. We identify the station agent associated with s_i , where $s_i \in S$, with $sa_i \in SA$ (the set of station agents), and the junction agent associated with j_k , where $j_k \in J$, with $ja_k \in JA$ (the set of junction agents).

Each train $t \in T$ is associated with a train agent $ta \in TA$ as soon as it is created in the IR system. Train agents are identified by ta subscripted, if needed, by a natural number i ranging from 1 to the maximum number of trains in IRN .

By using the same subscript for the agent and for the physical element of the IR system it is associated with, we can leave the bijective function that maps physical elements to agents implicit: throughout the paper, station agent sa_i will always be associated with station s_i , junction agent ja_i will be associated with junction j_i , and train agent ta_i will be associated with train t_i .

Fluent and Persistent Properties of the IR System

Properties of the IR system can be expressed by fluents (functions whose returned value changes over time), and by persistent functions which state structural properties of the system, not subject to changes over time. We assume that time is discrete and is represented by the number of time units that elapsed from a conventional starting instant. We express the time argument of fluents as a subscript of the function name for readability.

For each function, we write in square brackets and bold font the name of the only agent in the system which knows the function's value on the given arguments. This is a relevant aspect for emphasizing that data are really decentralized, and hence computation can be decentralized as well.

Besides all the standard operations on multigraphs (adjacency list, degree of a vertex, and so on), the only persistent function representing a relevant aspect of the IR physical network is

- $max_capacity : S \rightarrow \mathbb{N}$
 $max_capacity(s_i)[\mathbf{sa}_i] = n$ iff station s_i can host at most n trains. This information is only available to station agent sa_i .

Fluents representing IR's features are

- $current_capacity : \mathbb{N} \times S \rightarrow \mathbb{N}$
 $current_capacity_t(s_i)[\mathbf{sa}_i] = n$ iff station s_i has room for n more trains at time t . This information is only available to station agent sa_i .
- $running_on : \mathbb{N} \times E \rightarrow T^*$
 $running_on_t(e_{(i,j,k)})[\mathbf{sa}_i] = t_n \dots t_1$ iff t_n, \dots, t_1 are the trains currently running on edge $e_{(i,j,k)}$, in the order they appear in the string $t_n \dots t_1$ (namely, t_1 is the first train that left the station and it is the most distant from it, whereas

t_n is the last train that left it, and it is the closest). This information is only available to the station agent sa_i in charge of the station from which the edge exits.

- $is_free : \mathbb{N} \times J \rightarrow Bool$
 $is_free_t(j_i)[\mathbf{ja}_i]$ is true if junction j_i is free at time t , and false otherwise. This information is only available to junction agent ja_i .

As far as trains are concerned, persistent functions representing their official schedule and technical features are described below. Apart from the train's schedule, that is public, the only agent that possesses information about train t_i , is ta_i .

- $route : T \times \mathbb{N} \rightarrow S$
 $route(t_i, ind)[\mathbf{ta}_i] = s$ iff the ind -th station in t_i 's scheduled path is s (ind ranges between 1 and the maximum number of stations that t_i is expected to traverse).
- $scheduled_arrival : T \times S \rightarrow \mathbb{N}$
 $scheduled_arrival(t_i, s_j)[\mathbf{sa}_j] = n$ iff n is the time instant when t_i should arrive in s_j according to the planned schedule.
- $max_speed : T \rightarrow \mathbb{R}$
 $max_speed(t_i)[\mathbf{ta}_i] = r$ iff the maximum allowed speed for t_i is r , expressed in some suitable speed unit measure.
- $stop_value : T \times S \rightarrow Bool$
 $stop_value(t_i, s)[\mathbf{ta}_i]$ is true if t_i will stop in station s and false otherwise.

The fluents modeling train's features are

- $current_position : \mathbb{N} \times T \rightarrow (V \cup E) \times \mathbb{R}$
 $current_position_t(t_i)[\mathbf{ta}_i] = (v/e, r)$ iff t_i is currently either on vertex v , in which case r is 0, or on edge e , in which case r is the distance from the edge origin expressed in a suitable distance measure unit.
- $current_speed : \mathbb{N} \times T \rightarrow \mathbb{R}$
 $current_speed_t(t_i)[\mathbf{ta}_i] = r$ iff at time t , t_i is running at speed r , expressed in a suitable speed measure unit.

Finally, two fluents involve both trains and stations:

- $expected_waiting_time : \mathbb{N} \times T \times S \rightarrow \mathbb{N}$
 $expected_waiting_time_t(t_i, s_j)[\mathbf{ta}_i, \mathbf{sa}_j] = n$ iff n is the amount of time units that t_i should wait in station s_j .
- $expected_arrival : \mathbb{N} \times T \times S \rightarrow \mathbb{N}$
 $expected_arrival_t(t_i, s_j)[\mathbf{ta}_i, \mathbf{sa}_j] = n$ iff the time instant when t_i is expected to actually arrive in s_j , is n .

4 Tackled Problems and Proposed Solutions

In order to decide which train should access a resource first (the right to cross a station or a junction, the right to move on an edge), we introduce the notion of payoff of a train t_i that wants to enter a station s_j at time t .

$$\begin{aligned} \text{payoff} &: \mathbb{N} \times T \times S \rightarrow \mathbb{R} \\ \text{payoff}_t(t_i, s_j) &= \text{deviation_from_schedule}_t(t_i, s_j) * \text{deviation_cost} + \\ &\quad \text{expected_waiting_time}_t(t_i, s_j) * \text{waiting_cost} \end{aligned}$$

where

- $\text{deviation_from_schedule}_t(t_i, s_j) = |\text{expected_arrival}_t(t_i, s_j) - \text{scheduled_arrival}(t_i, s_j)|$ is the absolute value of the difference between the time when t_i should enter s_j according to its scheduled timetable, and the time when it is actually expected to enter it.
- deviation_cost is the penalty for a 1 time unit deviation from the scheduled timetable.
- $\text{expected_waiting_time}_t(t_i, s_j)$ is a value that we compute by exploiting the MAX-SUM algorithm [16, 17], as described in [12].
- waiting_cost is the penalty for a 1 time unit expected waiting in a station.

In our model, $\text{waiting_cost} > \text{deviation_cost}$, and both can be established by the system's administrator.

The same notion of payoff can be defined for junctions, instead of stations. Due to space limitations, we cannot give here more details about the MAX-SUM algorithm. We only point out that this algorithm uses the current delay of a train to increment the train's payoff: in this way, the payoff increases with the increase of the delay, and train starvation is avoided because sooner or later the train will reach a payoff higher than that of the other conflicting trains, getting the right to access a shared resource. Payoffs are used to resolve conflicts on shared resources according to the following criteria:

1. if more trains want to access the same resource at the same time, the resource will be assigned to the train with higher payoff;
2. a station agent associated with a station with few available platforms, can decide not to authorize a train to enter the station if its payoff is lower than the payoff of other trains that want to enter the station.

Now we can discuss the IR problems we faced and how we solved them.

Problem 1: if two trains are running on the same directed edge (obviously in the same direction) and their distance becomes lower than a critical distance cd , then a collision might take place.

When a train t_i leaves a station s_j on edge e , the station agent sa_j tells the train agent ta_i the identifier k of the train running ahead of it on edge e . In this way, ta_i can contact ta_k and coordinate with it in order to avoid collisions. The coordination is based on a simple rule: if there is a risk of collision computed based on the current speed and position of both trains, and the train which is ahead can accelerate (it is not running at its maximum allowed speed), than it does accelerate. Otherwise, the train which is behind decelerates. When the train which is ahead reaches the next station or leaves that edge, it informs both the station from which it departed and the train which is behind it. The station agent can remove that train from the list of trains running on e by updating the value of $\text{running_on}_t(e)$.

The proposed solution works only if the change of speed of one train removes the risk of collision with the train which is ahead (resp. behind) without introducing a collision risk with the train that is behind (resp. ahead). In other words, we are assuming that if t_3 , t_2 and t_1 are running on the same edge and t_2 has to accelerate to avoid a collision with t_3 , this acceleration will not cause a collision with t_1 . This is a very strong assumption and we are extending our negotiation algorithm in order to overcome it. The extended algorithm will fire a message exchange with the (at most two) adjacent trains, whenever a train changes its speed. In this way changing the speed propagates the collision risk, but also the communication among agents to prevent it.

Problem 2: if the number of trains hosted by a station is equal to its maximum capacity, then no more trains can enter it otherwise a collision might take place. When a train t_i wants to enter a station s_j , ta_i sends a message to sa_j asking a permission to enter. If $current_capacity_t(s_j) \geq 0$, then sa_j answers to ta_i that t_i can enter the station, otherwise t_i will need to stop and wait that one train leaves s_j . We assume that t_i can stop outside s_j without creating any collision with the trains behind it, if any, running toward s_j . In case there is only one bidirectional edge entering s_j (recall that IRN is a mixed graph), it must be used both by t_i for entering s_j and by the trains already in s_j for exiting, leading to a deadlock. This is a borderline situation that we assume will not take place. We will relax these constraints in the forthcoming version of the negotiation algorithm.

In case of two simultaneous requests involving one train that would stop in the station and another that just needs to traverse it, then the train that will not stop in s_j gets the right to enter. If the trains have the same *stop_value*, the agent with higher payoff will be allowed to enter the station.

Problem 3: if two trains are leaving a station on the same edge at the same time (modulo a small time difference), a collision might occur.

To resolve this problem, when a station agent allows a train t_i to leave on the edge e , it must not allow other trains to leave on e until t_i moves of at least cd (the critical distance). In this situation, the train that should arrive to the next station first is allowed to move on the edge first.

Problem 4: when a train t enters a junction or a station where it should not stop, planning to exit on edge e , and another train is either already moving on e within the critical distance cd , or is planning to move on it, then a collision might occur.

This situation can be faced considering two cases:

1. t_i wants to cross a junction or a station v (without stopping on it), but the edge e where it plans to move is already occupied by t_j whose distance from v is lower than cd . The only action that the agent associated with v can do is to prevent t_i from entering v until the distance between t_j and v becomes greater than cd .
2. t_i wants to cross a station s exiting on edge e , and t_j is waiting in s and is ready to leave it, moving on edge e . Since trains that do not stop in a station

have a higher priority w.r.t. those that stop, t_j is asked to wait, and t_i is allowed to cross s and exit on e .

In both cases we assume that stopping a train on an edge outside a station or a junction does not cause a collision with the train behind that is arriving on that edge. Also this assumption will be removed in the refinement of the current algorithm.

Problem 5: if a bidirectional edge exists between two vertices v_i and v_j and t_h would like to move from v_i to v_j , while t_k would like to move from v_j to v_i , then a collision might occur.

If both trains are still waiting in their stations, this problem is solved by allowing the train with higher payoff to use the bidirectional edge between v_i and v_j first. If one train is already moving on the bidirectional edge, the other train cannot move on it until the edge becomes free again.

Problem 6: given that avoiding collisions is the first goal of the IR system, how could we minimize the total delay of trains running in the IR network?

The delay optimization algorithm we have selected for solving the last problem is the MAX-SUM algorithm described in [16, 17], with the payoff introduced at the beginning of this section. Since the MAX-SUM algorithm is not new, and its instantiation to the IR system has been described in [12], we do not enter into its details here. By adopting the MAX-SUM algorithm, our system is guaranteed to converge into an optimal solution, which minimizes the individual train delay as well as the total system delay, and which additionally reduces the communication and computation cost also.

5 Experiments

We simulated our MAS using NetLogo because of its support for the rapid development of a graphical interface by which an IR administrator could visualize the whole network and the movement of each individual train. The NetLogo program needs three text files as an input:

- i) Path.txt, which contains information on the modeled stations
- ii) Time_table.txt, which contains the trains' scheduled timetable
- iii) Route.txt, which contains a list of stations that a train must follow in its journey (namely, the values of $route(t_i, ind)[\mathbf{ta}_i] = s$, for each ind from 1 to the number of stations that t_i must traverse). As NetLogo does not support any message passing protocol between agents, we implemented it by ourselves.

In the next sections we discuss our case studies and the results we obtained by the simulations we executed.

Case Studies

The data we used for our experiments have been provided to us by the Eastern India Railway officials. We simulated a sub-network consisting of 42 Km of track with 11 stations, whose features are shown in Table 1.

Table 1. Stations in the simulated *IR* sub-network

Station id	Num. of platforms	Adj. stations (with distance in Km)
1	5	2 (8.9)
2	4	3 (7.6), 1 (8.9)
3	6	4 (7.5), 7 (3.0), 2 (7.6)
4	4	5 (12.5), 3 (7.5)
5	4	6 (5.5), 4 (12.5)
6	7	5 (5.5)
7	2	8 (2.3), 3 (3.0)
8	2	9 (4.5), 7 (2.3)
9	2	10 (2.5), 8 (4.5)
10	2	11 (2.5), 9 (2.5)
11	2	10 (2.5)

The simulation involves 128 trains: 55 trains run from station 1 to station 6 and 55 trains run from station 6 to station 1. As station 11 lies in a very remote area, only 2 trains are running from station 1 to station 11 via station 3 and vice versa. Similarly, only 8 trains are scheduled from station 3 to 11 and 6 trains are running from station 11 to station 3.

We simulated: the existing non-automated system (without modeling the manual interventions and communication used in order to coordinate the train movements), the MAS where the solutions to the first five problems highlighted in Section 4 have been implemented, and the MAS where the sixth problem has been faced by means of the MAX-SUM algorithm.

1. Simulation of the non-automated system. In the first experiment, we simulated the existing IR sub-network with neither automation nor manual intervention.

2. Simulation of the proposed MAS, without delay optimization. In the second experiment, we simulated the same sub-network, with the same trains running over it as in the first experiments, but where coordination was guaranteed by the MAS described in the previous two sections.

3. Simulation of the proposed MAS, with delay optimization. In the third experiment, we simulated the same sub-network as in the first and second experiments, but where delay is minimized thanks to the MAX-SUM algorithm. With respect to the previous experiments, we changed the number of trains: we created six different configurations with 50, 100, 150, 200, 150 and 300 trains by introducing some random delay and simulated them in NetLogo.

Results

1. Simulation of the non-automated system. The result of the first experiment was that trains got 3.8 hours of cumulative delay in 24 simulated hours (second column of Table 2) and 4 collisions took place. Of course (and luckily!), in

the real system such collisions did not take place because of the manual coordination and communication among the IR staff. However, we could not really compare our simulated automated system with the (simulated) actual one, since the IR staff human interventions are so complex and dependent on a large set of environmental variables that they are not reproducible within a simulation. Anyway, the IR staff main objective is avoiding collisions, with little care towards reducing trains delay. Hence, as far as delay reduction is concerned, the actual IR system is almost close to the non-automated one that we simulated and our comparison between the automated and non-automated systems discussed in the two paragraphs below is realistic.

2. *Simulation of the proposed MAS, without delay optimization.* The result was that trains got 1.2 hours of cumulative delay in 24 simulated hours (third column of Table 2) and no collisions took place.

3. *Simulation of the proposed MAS, with delay optimization.* The results of the six experiments we run, with different number of trains, are shown in Table 3 and demonstrate that the MAX-SUM algorithm achieves its goal of optimizing the overall delay.

Table 2. Delay Comparison between non-automated system and MAS (in hours)

Simulated hour	Delay in non-autom. system	Delay in proposed MAS
0	0	0
2	0.125	0
4	0.27	0
6	0.35	0
8	0.48	0.02
10	0.56	0.053
12	0.67	0.18
14	0.75	0.2
16	0.83	0.22
18	0.91	0.26
20	0.98	0.32
22	0.99	0.33
24	3.8	1.2

Discussion

As shown in Figures 1, where the red line (dark grey in b/w images) represents the non-automated system and the green line (light grey in b/w images) represents our MAS without delay optimization, the automation of the IR system according to our model and algorithms gives an high advantage in terms of global delay with respect to a non-automated system. Furthermore, no collisions happen.

Table 3. Delay Comparison between MAS without and with MAX-SUM (in minutes)

Num. of trains	Delay without MAX-SUM	Delay with MAX-SUM
50	0.33	0.33
100	2.33	2
150	9.83	4.67
200	32.83	4.5
250	39.5	5.17
300	47.33	28

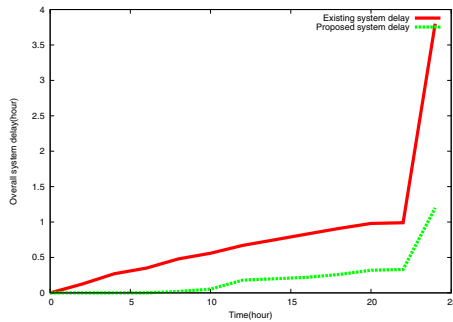


Fig. 1. Non-automated vs automated system: delay comparison

Figure 2 shows that the gain in reduced delay when using the MAX-SUM algorithm increases when the number of trains grows up (red line represents the MAS without MAX-SUM and the green line represents the MAS where MAX-SUM has been exploited).

Despite the strong assumptions that we made, the results we obtained in our attempt to model and simulate the huge and complex Indian Railways system are very encouraging both in the reduction of the trains' delay and, most important, in the complete avoidance of collisions.

6 Conclusions and Future Work

In this paper we have presented a model and negotiation algorithms for automatizing the IR signaling and control tasks. Given the features of the IR system, modeling it as a MAS was a very natural choice, and using the NetLogo widespread MAS simulation tool for carrying out our experiments was a further natural consequence of the selected agent-based approach. The results of our experiments are very promising: the system is collision free and the overall delay is reduced. An implementation of the system in Jade⁵ has been designed and a first prototype has been implemented. As for the FYPA system, the exploitation

⁵ jade.tilab.com/, accessed on May, 2013.

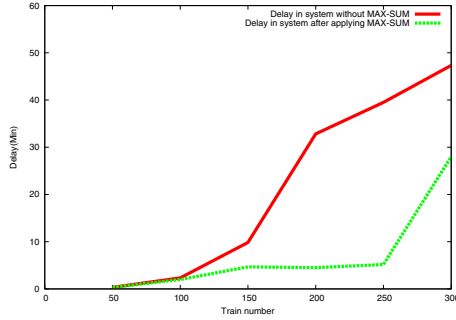


Fig. 2. Delay reduction after the application of MAX-SUM

of Jade gives the possibility of integrating real databases and existing software components by building a Java bridge towards them, which is usually possible (even if not always easy).

Although we are still far from thinking that IR could adopt our solution, the integration of our MAS on top of the existing IR framework would be feasible and could be done with a low capital investment because the MAS could be implemented using open source software as Jade and could be installed in the existing railway control rooms. The major cost of this technological shift would be due to the improvement of the IR information technology and physical infrastructures to get more reliable and real-time data from the network, which is a requirement for the adoption of our solution. The estimation of this investment is out of our competencies and can be only done by the IR.

The goals we plan to pursue in the future include:

- collecting real data on the railway traffic during some days (delays and, we hope none, collisions), to make a comparison between the solution proposed by our system, simulating the same days, with the actual human solution, instead of using the non-automated system as a reference;
- eliminating the strongest assumptions we made in our model, in order to make our system closer to the real scenario;
- incorporating some intelligent rule engine like JESS⁶ to make the agents reasoning and coordination activities smarter than now;
- integrating an ontology representing the domain to better model communication semantics.

References

1. Axelrod, R.: Advancing the art of simulation in the social sciences. *Complex* 3(2), 16–22 (1997)
2. Bandini, S., Manzoni, S., Vizzari, G.: Agent based modeling and simulation: An informatics perspective. *Journal of Artificial Societies and Social Simulation* 12(4), 4 (2009)

⁶ <http://herzberg.ca.sandia.gov/>, accessed on May, 2013.

3. Bhardwaj, A., Ghosh, S., Dutta, A.: Modeling of multiagent based railway system using BDI logic. In: International Conference on Future Trends in Computing and Communication (to appear, 2013)
4. Böcker, J., Lind, J., Zirkler, B.: Using a multi-agent approach to optimise the train coupling and sharing system. *European Journal of Operational Research* 131(2), 242–252 (2001)
5. Briola, D., Mascardi, V.: Design and implementation of a NetLogo interface for the stand-alone FYPA system. In: Proceedings of the WOA 2011, pp. 41–50 (2011)
6. Briola, D., Mascardi, V., Martelli, M.: Intelligent agents that monitor, diagnose and solve problems: Two success stories of industry-university collaboration. *Journal of Information Assurance and Security* 4(2), 106–116 (2009)
7. Briola, D., Mascardi, V., Martelli, M., Caccia, R., Milani, C.: Dynamic resource allocation in a MAS: A case study from the industry. In: Proceedings From Objects to Agents Workshop, WOA 2009 (2009)
8. Chen, B., Cheng, H.H.: A review of the applications of agent technology in traffic and transportation systems. *Trans. Intell. Transport. Sys.* 11(2), 485–497 (2010)
9. Cuppari, A., Guida, P.L., Martelli, M., Mascardi, V., Zini, F.: Prototyping freight trains traffic management using multi-agent systems. In: Proc. of IEEE International Conference on Information, Intelligence and Systems. IEEE (1999)
10. Fischer, K., Kuhn, N., Muller, H.J., Muller, J., Pischel, M.: Sophisticated and distributed: The transportation domain. In: Proceedings of the Ninth Conference on Artificial Intelligence for Applications, page 454 (1993)
11. Fischer, K., Kuhn, N., Muller, J.: Distributed, knowledge-based, reactive scheduling of transportation tasks. In: Proceedings of the Tenth Conference on Artificial Intelligence for Applications, pp. 47–53 (1994)
12. Ghosh, S., Dutta, A.: Multi-agent based railway track management system. In: Proc. of 3rd IEEE International Conference on Advance Computing & Communication, pp. 1408–1413 (2013)
13. Vierke, G., Bürckert, H.-J., Fischer, K.: Transportation scheduling with holonic mas - the teletruck approach. In: Proceedings of the 14th European Meeting on Cybernetics and Systems Research (1998)
14. Lind, J., Fischer, K., Böcker, J., Zirkler, B.: Transportation scheduling and simulation in a railroad scenario: A multi-agent approach. In: IN PAAM 1999, pp. 325–344. Springer (1999)
15. Proença, H., Oliveira, E.: MARCS multi-agent railway control system. In: Lemaître, C., Reyes, C.A., González, J.A. (eds.) IBERAMIA 2004. LNCS (LNAI), vol. 3315, pp. 12–21. Springer, Heidelberg (2004)
16. Stranders, R., Farinelli, A., Rogers, A., Jennings, N.R.: Decentralised coordination of continuously valued control parameters using the Max-Sum algorithm. In: Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (2009)
17. Stranders, R., Farinelli, A., Rogers, A., Jennings, N.R.: Decentralised coordination of mobile sensors using the max-sum algorithm. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (2009)
18. Tsang, C.-W., Ho, T.-K.: Optimal track access rights allocation for agent negotiation in an open railway market. *IEEE Transactions on Intelligent Transportation Systems* 9(1), 68–82 (2008)

Towards a Platform for Testing and Developing Privacy-Preserving Data Mining Applications for Smart Grids

Andrew Koster¹, Gabriel de Oliveira Ramos¹,
Ana L.C. Bazzan¹, and Fernando Koch²

¹ Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS)
² IBM Research Brazil

Abstract. In this paper we analyse the trade-off between privacy-preservation methods and the quality of data mining applications, within the specific context of the smart grid. The use of smart meters to automate data collection is set to solve the problem of electricity theft, which is a serious concern in developing nations. Nevertheless, the unlimited use of data from smart meters allows for potentially private information to be discovered. There is a demand for methods to quantify the trade-off between privacy-preservation and quality of a classification model. We describe the research and development of an agent-based simulation platform to evaluate the balance between privacy-preservation mechanisms and methods for electricity theft detection. We have implemented a proof-of-concept model and validated it against real data collected from smart meters.

1 Introduction

A smart grid is an electricity and utilities grid instrumented to automatically collect and act upon information about the behaviour of suppliers and consumers. This technology aims to improve the efficiency, reliability, economics, and sustainability of the production and distribution of electricity. A common element in smart grids are the *smart meters*, used to collect information about household utilisation. Nonetheless, Cohen [9] alerts that “[smarter meters are] vulnerable to remote exploitation, viruses, worms, malicious upgrades, and all manner of other attacks”. Hence, the utilisation of such devices represent a concern to individual privacy, despite the obvious benefits to electricity companies.

The interest of this research is to mitigate the aforementioned concerns without compromising the rationale for applying smarter meters in the first place. To this end, we need to quantify the trade-off between (i) collecting detailed data through smart meters and (ii) concerns about privacy issues by individuals and groups. In particular we focus on the use of smart meters to detect electricity theft, a major problem in many developing nations, and one which information collection from smart meters can combat effectively.

In this paper, we introduce a framework for measuring the trade-off between privacy-preserving methods and data mining techniques for detecting electricity

theft. It encompasses an agent-based simulation model to (i) represent different trust and privacy profiles with regards to sharing information, considering that different people have different levels of trust and different preferences when it comes to what data is kept private, and; (ii) generate households' electricity load profiles, by applying publicly available data to create reference models, and concepts of social behaviour to create extended data sets that simulate community diversity. Then, we are able to apply probabilistic approach and data mining techniques to detect the expected behaviours in the different community setups, allowing us to measure the trade-off between privacy-preserving methods and approaches for collecting data to detect electricity thievery.

The outcome of this research provides new ways to understand the impact that the introduction of smarter grids will have in development communities. We are particularly interested in the situation of Brazil, where smarter grids are being implemented to help combat electricity theft. According to ANEEL, Brazil's national electricity agency, in some regions this problem compromises up to 25% of the total electricity production. The side-effects go beyond economical balance for energy companies, as ultimately this cost is shifted to consumers in terms of raising energy prices [16].

Moreover, this research is of interest to providers of smart grid technology. It allows for a better understanding of the social impact of applying the smart meters in different communities. In this context, we are building on the solutions and case scenarios in the *IBM Smart Grid* program [12]. This development envisages "a layer of intelligence throughout the grid to enhance system reliability and efficiency, improve management of supply and demand, optimize operations and streamline costs". We will contribute to this program with a layer of understanding about the impact and acceptability of the technology by different communities.

The paper is structured as follows. In Section 2 we discuss our motivation and present the related work in the areas of data mining methods for theft detection and privacy preservation methods. Section 3 describes our framework and Section 4 provides a proof-of-concept validation of this framework. The paper concludes with a discussion and an analysis of future work in Section 5.

2 Motivation and Related Work

The use of smart meters brings many benefits; one of which is the ability to detect and shut down electricity theft, but there are many other uses that benefit both electricity providers (such as the use of this data in forecasting electricity use) and end-users (time-of-use billing can lower overall expenses). However, the data collection that allows these uses is a double-edged sword. Load-monitoring allows the identification of specific electrical appliances, such as medical equipment, or baby monitors [14]. Over time, the data collected can be used to discover patterns of use, from which it can deduced whether a family is at home, or even when particular family members are at home. The European Union's Data Protection Supervisor recommends that, in addition to other measures, privacy-enhancing

technologies be made mandatory in the deployment of smart electricity meters [7]. While most users are not concerned with what their electricity provider can do with this information, the information is valuable. There are significant security concerns with the storage of such data, and there are no regulations in place to prevent the sale of the data to untrusted third parties.

There is a considerable amount of work on both privacy preservation for data mining techniques, as well as in the detection of electricity theft using smart meter technology. However, insofar as we know there is no work that attempts to quantify the trade-off between privacy and a specific knowledge extraction goal. We discuss some of the related work in both these separate domains below.

2.1 Detection of Electricity Theft

We emphasise that the data collection from traditional meters, which do not cause any privacy concerns, is not detailed enough to detect theft reliably through data mining techniques. For instance, Cabral and Gontijo [8] use so-called rough sets to derive rules that allow the classification of fraudulent customers, using data from traditional meters, and achieve an accuracy of 20%, which at the time was considered good. Smart meters, however, collect, and automatically send, far more fine-grained readings to the utilities companies, and using this data there are many novel approaches for detecting fraudulent activities. Kadurek et al [13] have proposed a methodology for automated detection of electricity theft, which does not require data collection and is performed in real-time at the substation, but they do not present data on how well it works, particularly in a market where electricity theft is more prevalent than the Netherlands (where they deploy their prototype).

Most state-of-the-art work in the area approaches the problem with data mining techniques. For instance, Nagi et al. use Support Vector Machines (SVMs) to detect customers with irregular consumption patterns, which are associated with fraudulent profiles [15], and Ramos et al. use a graph-based classifier, optimum-path forest (OPF) in order to detect NTLs [18]. It is not our intention to give a complete overview of the techniques used: rather, it should be clear that such methods require extensive sets of detailed information to distinguish between larcenous and honest households. Nagi et al. use a set of data from 500 households, whose electricity readings were recorded hourly for two years. Ramos et al. used data that was collected at 15 minute intervals. This kind of data can be used to uncover privacy-sensitive information and thus some form of privacy protection must necessarily be employed.

2.2 Privacy Protection in Smart Grid Technology

We are not the first to recognise the need for privacy-preserving methods for use in the smart grid. Erkin et al. survey a number of approaches designed specifically to prevent revealing sensitive information [11]. These methods all use secure signal processing techniques to encrypt an individual household's load data.

Because of specific properties of the cryptographic methods used, the aggregation can be performed on the encrypted data, meaning that when the data is decrypted, individual households' load data cannot be retrieved. While this is effective if the use of the smart meter is primarily to build predictive models for load balancing or for adapting the price in a time-of-use billing mechanism, it denies any possibility of using the data to discover fraudulent individuals.

A more promising alternative is to build on the privacy-preservation techniques in data mining. Aggarwal and Yu [1] identify various methods for this: randomisation, the k -anonymity model, distributed privacy-preserving data mining and downgrading the classifier effectiveness. We intend to test the functioning of a number of these methods. In particular randomisation seems promising: by adding random noise to the data it may be possible to sufficiently hide privacy-sensitive information while still allowing fraudulent individuals to be detected. One concern with this is obviously that real data must still be sent for the purpose of billing, but this is only needed once per billable period.

3 Methodology

The main aim of this paper is to lay out a clear methodology for evaluating the inevitable trade-off between detecting electricity theft and preserving households' privacy. While our long-term goal is to develop methods that allow theft-detection algorithms to maintain an adequate level of performance while preserving an adequate level of privacy, we need a method for making explicit what an "adequate level" is in both these cases and measure the trade-off explicitly. While we focus on theft detection as the principal application for smart meters in this paper, we wish to emphasise that the same techniques are usable for other applications of machine learning techniques on data from smart meters, such as load prediction.

The two main problems in measuring the trade-off are:

1. Testing different preferences when it comes to privacy.
2. Quantifying the trade-off between privacy-preservation and accuracy of theft detection.

In Section 3.1 we propose an agent-based simulation to solve the former problem. In Section 3.2 we present a statistical method for solving the latter.

3.1 Simulating Households' Electricity Use

Different people have different requirements when it comes to privacy. Some households may not mind providing any amount of information, regardless of what sensitive details it reveals. On the other side of the spectrum are very private individuals who are uncomfortable giving out any more information than they do currently. Both are probably minority groups, with the majority of people willing to reveal some information as long as it does not reveal sensitive

information, and enough guarantees are given that the data will not be further distributed (willingly or due to security leaks) without their permission.

This all is closely related to trust. For someone to feel comfortable giving out (potentially) sensitive information, there must be a trust relationship that this information will not be mistreated. Different people have different levels of trust and different preferences when it comes to what data is kept private. This can be simulated using a multi-agent system in which agents have different trust and privacy profiles with regards to sharing information. This also allows for distributed privacy methods to be tested, where network proximity (using either a physical network or a social one) is used to do some preprocessing in order to preserve users' privacy at the global level. However, then trust is needed at a local level as well; between households and not just between the household and the electricity provider.

Furthermore, the agent-based simulation can be used to generate the load profiles. This alleviates the problem of obtaining real data with sufficient detail. The load data available publicly, such as that made available by the Remodece project [10], is only from honest households. Such data could be used to test privacy-preserving methods, but not to distinguish between larcenous and honest households. On the other hand, data such as the load profiles available in the work by Nagi et al. [15] makes a clear distinction between the profiles of honest and fraudulent households, but there is not enough detailed information available to discover any privacy-sensitive information. As such we propose a simulation-based approach in which each agent represents a household with a specific privacy profile and generating an electricity load profile as in the work by Paatero and Lund [17] or Armstrong et al. [3]. Larcenous agents, however, deviate from the profile and instead are given profiles based on the profiles of fraudulent agents in Nagi et al.'s work. An initial step to verify this simulation-based approach is described in Section 4.

3.2 Measures

The assumption in the various theft-detection algorithms is that a detailed load profile for each household is available, which allows an accurate distinction between profiles of larcenous and lawful households. However, such detailed information allows for infringement on users' privacy.

In this section we provide a framework for measuring the trade-off between privacy-preserving methods and data mining techniques for detecting electricity theft. While privacy is generally regarded in absolute measures (such as in the encryption approach discussed in Section 2, this is not the case here: we want to know how well a privacy-preservation method hides sensitive information, with respect to the original data. For instance, if a piece of private information is already hidden, then it is not necessary to use a privacy-preservation method. We will quantify how well a user's privacy is preserved using some privacy-preservation technique with such a relative measure.

Similarly, the measure for how well the difference between larcenous and lawful individuals can be learned using machine learning methods, is also relative: we

want to know how much using a privacy-preservation method impacts a ML algorithm’s functioning with respect to the original dataset. By quantifying this property as well, we can quantify the trade-off between the two different interests, and this gives us some idea of whether it is a worthwhile approach or not.

Quantifying Privacy Preservation. There are a number of different ways of quantifying privacy preservation, designed for different uses [5]. Most take a probabilistic approach and see how much “harder” it is to guess the right answer after performing the privacy-preserving operation. One of the most prominent, presented by Agrawal and Aggarwal [2], perform this quantification by using the conditional entropy. Using the entropy $H(A)$ of a random variable A , they define the privacy inherent in that variable as $\Pi(A) = 2^{H(A)}$. The privacy *lost* between two random variables is given in terms of conditional entropy. If we consider the data as a random variable A with domain Ω_A , and the privacy-preserved data B , the conditional entropy is given as follows:

$$\begin{aligned}
 H(A|B) &= \int_{y \in \Omega_A} f_B(y) \cdot H(A|B = y) dy \\
 &= - \int_{y \in \Omega_A} \int_{x \in \Omega_A} f_{A,B}(x, y) \cdot \log_2(f_{A|B=y}(x)) dx dy
 \end{aligned}
 \tag{1}$$

Where we assume the privacy preservation operation does not change the domain of the random variable. f_B is the probability density function for variable B . Similarly, $f_{A,B}$ is the density function for the joint probability of A, B and $f_{A|B=y}$ the density function for the conditional probability.

Using the conditional entropy, Agrawal and Aggarwal define the “fraction of privacy loss”, or the amount of privacy that is lost with regards to A by knowing B as $\mathcal{P}(A|B) = 1 - \Pi(A|B)/\Pi(A)$. We use this in a slightly modified form as our privacy preservation measure (*PPM*):

$$PPM(A|B) = H(A|B)/H(A)
 \tag{2}$$

Rather than using their measure of privacy, we use the entropy directly. The advantage of this measure over Agrawal and Aggarwal’s measure is that our measure is 0 if A is entirely determined by B , whereas the original measure is $1 - 1/\Pi(A)$, which is only 0 if the entropy of A is 0. Our measure is undefined in this trivial case and we define it separately as $PPM(A|B) = 0$ if $H(A) = 0$. The original dataset has no privacy-sensitive properties to preserve. If A and B are independent, then both the measures are 1.

Example. We illustrate this with an example for discrete random variables. Assume a dataset collected from 500 homes, 450 with at least one child and 50 without children. We thus have $H(A) = \sum_{x \in \{c, \neg c\}} -p(x) \cdot \log_2(p(x)) = 0.47$ bits.

Now assume we have a privacy preserving function that randomly labels 200 of the homes with at least one child as childless, resulting in set B . We have the following probabilities: $P(A = c|B = c) = 1$, $P(A = c|B = \neg c) = 4/5$, $P(A =$

$\neg c|B = c) = 0$ and $P(A = \neg c|B = \neg c) = 1/5$. We thus have $H(A|B) = 0.36$ and $PPM(A|B) = 0.77$. This corresponds with what we would expect from a machine learning algorithm: if we train an algorithm on set B , it will learn to misclassify a large number of families with children as being childless and will thus be inaccurate on the original set A . Because the actual accuracy depends on specifics of the machine learning algorithm, it makes more sense to specify this in terms of relative entropy than in terms like precision or recall.

A second privacy-preserving function creates set C by simply removing 200 homes where at least one child is childless. We then have the following probabilities $P(A = c|C = c) = 1$, $P(A = c|C = \neg c) = 0$, $P(A = \neg c|C = c) = 0$ and $P(A = \neg c|C = \neg c) = 1$. This results in $H(A|C) = 0$ and also a $PPM(A|C) = 0$. It is clear why: if we learn which of the families in C have children, we can use that same classifier on set A and expect it to be accurate.

Quantifying Theft Detection. In contrast to the quantification of privacy-preservation, which we want to do independent of the specifics of the machine learning algorithm, for theft detection we are particularly interested in how well the machine learning algorithm performs. There are a number of measures that are traditionally used to quantify the functioning of machine learning algorithms, most notably *precision* and *recall* [4].

Precision can be seen as the probability that a positively classified household is a true positive. Recall, on the other hand is the probability that a true positive is correctly classified as positive. These two measures are combined into the F_β measure as follows:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} \quad (3)$$

This can be interpreted as a weighted average (the harmonic mean) of precision and recall, with weighting factor β . If $\beta > 1$, then recall is given more importance than precision and $\beta < 1$ the reverse. Common values for β are 0.5, 1 and 2. We suggest to use 2, because in detecting theft we are interested in casting a fairly wide net: every positive hit will need to be verified in any case to decide whether legal action should be taken. Of course, the net shouldn't be too wide or it is useless, so precision of the method should not be ignored.

Note that the F_β measure is always between 0 and 1. If either recall or precision are 0, then F_β is 0, and if both are 1, then F_β is also 1. Anything else results in an intermediate value. The measure is undefined if $F_\beta(A) = 0$.

However, we are not interested in the absolute performance of a machine learning method, but rather in the relative performance on a dataset that has been modified by a privacy-preserving method with respect to the original performance. As such we consider A the dataset before privacy-preserving measures and B after, with corresponding performance measures of a machine learning method $F_\beta(A)$ and $F_\beta(B)$ respectively. The theft detection measure is then:

$$TDM(A, B) = F_\beta(B)/F_\beta(A) \quad (4)$$

While this ratio is not limited to the $[0, 1]$ range, it is only greater than 1 if the privacy-preserving method actually improves the performance of the theft detection method. If we are afraid of this happening we can simply take the minimum value between the TDM as calculated in Eq. (4) and 1. We define $TDM(A, B) = 1$ if $F_\beta(A) = 0$, because if the recall and precision of the machine learning algorithm are both 0 on the original data set, no privacy-preservation method is going to make the method perform any worse.

Measuring the Trade-off. We now have two measures that can be seen as a way of measuring how well the privacy-preserved data performs with regards to the original. If the PPM is 0, then the modified dataset preserves privacy equally well as the original. Similarly if the TDM is 1, the dataset allows for equally good theft detection as the original. It is necessary to be very careful in comparing these two values, because strictly speaking they are not comparable: a 0.1 increase in privacy protection does not mean the same as 0.1 increase in theft-detection. Nevertheless, the measure $PPM(A|B) + TDM(A, B)$ can give a rough estimate of how well we are accomplishing the trade-off between the two conflicting goals. $PPD(A|A) + TDM(A, A) = 1$, so if the measure drops below 1 this could indicate that we are losing performance: the amount of privacy we have gained, as quantified by the PPM is less than the accuracy in theft detection we have lost, as measured using the TDM . Similarly if the measure is greater than 1 it can indicate that the loss in accuracy is offset by a greater gain in privacy preservation.

In future work we aim to evaluate the usefulness of this measure simultaneously with privacy preservation methods.

3.3 Putting It All Together

The simulator, as described in Section 3.1 simulates the data that could be collected by an electricity provider, where different households have different privacy profiles. In addition, the simulator can generate the data with full access. This gives us our two sets A and B , generated in a way that respects individual wishes for privacy. The measures of Section 3.2 can then be used as an indicator of whether we can accurately detect larcenous households and whether the data has been protected against privacy infringements. This can guide us in our research into better methods for preserving users' privacy, which can be codeveloped with datamining techniques for detecting electricity theft in privacy-protected data, and possible other applications of data from smart meters.

4 Simulation and Experimentation

We have implemented a preliminary model to demonstrate the viability of the approach detailed in the previous section. This prototype model does not implement the full framework as presented in the previous section, because we do not incorporate privacy-sensitive information into households' load profiles.

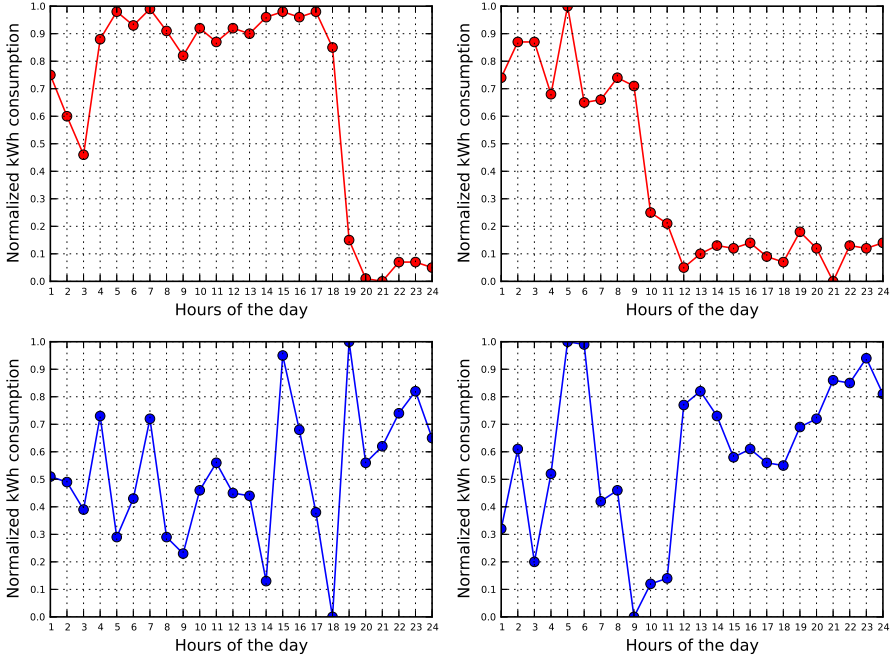


Fig. 1. The four (normalised) profiles from Nagi et al. [15], used as templates for modelling load profiles. The two top profiles (in red) are typical profiles of fraudulent households and the bottom ones (in blue) are of honest households.

Nevertheless, this prototype serves as a proof-of-concept for the model, and we demonstrate how the generation of honest and fraudulent households works. In Section 4.2 we validate the model empirically, but first we explain how it works.

4.1 Simulation Setup

In this prototype implementation, we model a neighbourhood, consisting of N households, with a percentage F of these households fraudulent. We assume that every smart meter reports its energy consumption on each hour period, and as stated above, do not yet generate privacy-sensitive information. The energy consumption, or load profile, that is generated, is based on the load profiles as presented in Nagi et al. [15], who present four typical load profiles from their set of historical consumption data in Malaysia (see Figure 1): their data set includes load profiles of fraudulent households and they disclose two typical profiles each for honest and fraudulent households. These load profiles serve as templates for generating the load profiles in our simulation.

We generate individual households' load profiles by starting with one of the templates and adding Gaussian noise, with a standard deviation σ to each of the datapoints. The choice between the four templates is decided by the percentage

F of fraudulent households and the percentage T of households using the profiles from the first column (and thus $1 - T$ using the second column).

All the experiments in the following section are run with the following parameters, unless stated differently. $N = 400$: this is the approximate size of the dataset Nagi et al. used, as well what we estimate is a typical size for a favela in Rio de Janeiro, based on recent census data. $F = 0.15$: LIGHT, the primary electricity provider in Rio de Janeiro, estimates that 15% of the electricity is lost in non-technical losses¹. This number also corresponds with ANEEL’s report, as well as Nagi et al.’s data. $T = 0.5$: we have no reason to favour one profile over another, so choose a uniform distribution. Finally, we determine the value for σ empirically in the next section, by comparing the simulated data for honest households to a dataset of load profiles from real households.

We do not incorporate privacy-sensitive information in the simulated households in this iteration of the paper, and thus do not use an agent-based model. Nevertheless, this model serves as a proof-of-concept for the method and we expect to extend the simulation with privacy-sensitive information, as described in Section 3.1.

4.2 Experimentation

In order to validate the simulation method, we perform two experiments. In the first, we show that the profiles for honest households, generated using our model, are similar to the load profiles of real households, collected in the Remodece project [10]. In the second, we show that machine learning algorithms are capable of detecting electricity theft in our simulated neighbourhood.

Experiment 1: Realism of Generated Profiles. In this experiment we show that the method for generating honest profiles generates profiles that are realistic and to determine what σ to use for the next experiment in order to keep the data as realistic as possible. For this, we compare the profiles generated for honest households to real data gathered from, insofar as anybody knows, honest households. The Remodece project collected data from smart meters in a number of countries in Europe, and we compare our generated load profiles against the load profiles in these datasets.

Because there is a high amount of variation between different households’ use of electricity, it is not possible to compare load profiles with each other “directly” (even two randomly chosen profiles from the same data set may show no correlation at all with each other). We therefore compare the datasets statistically: for every hour, we test whether the real data and simulated data have similar distributions. Because neither our real data, nor the simulated data, are normally distributed (the simulated data follows a bimodal distribution: we add noise to two different template profiles), we need to use a non-parametric test. Two choices stand out, the Mann-Whitney U test and the Kolmogorov-Smirnov

¹ <http://www.relatoriolight.com.br/energia-cintica/distribuio/qualidade-na-distribuio?lang=en-US>

test. Both are applicable and test for slightly different things. The latter is generally less powerful for deciding whether two populations are similar, but has the added benefit of detecting differences in the shape of the distributions in addition to differences in the average rank. We thus simply use both and check the simulated data for different values of σ against the real data. We average the p -values for each test over the hours in the day, and the results can be found in Table 1.

Table 1. Average p -values of the Mann-Whitney U test and Kolmogorov-Smirnov test between simulated and real data. The minimum values are bolded.

σ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
MW U test	1.7e-2	5.0e-2	4.4e-2	2.6e-2	7.3e-3	4.6e-2	3.4e-2	3.2e-2	1.8e-2	6.4e-2
KS test	8.5e-3	4.2e-3	1.0e-2	1.9e-3	9.2e-4	9.2e-4	2.5e-3	2.9e-3	1.8e-3	2.6e-2

As expected given the characteristics of the test, the p -value for the Kolmogorov-Smirnov test is always smaller than for the Mann-Whitney U-test, and if we accept the hypothesis that the two distributions are similar at $p < 0.05$, this hypothesis is never rejected by it, while the Mann-Whitney U test does reject it for a number of values of σ . However, for both tests the p -value is smallest at $\sigma = 0.5$, which we will adopt in Experiment 2.

The results of this experiment seem to indicate that adding Gaussian noise to some template profiles allows us to generate realistic profiles, however we are hesitant to conclude this. Firstly, due to the large variation in profiles we have only been able to perform statistical tests per timestamp, rather than test whether the profiles truly are correlated. Secondly, it is possible that the template profiles we used happened to be “good” templates for the Remodece dataset, and this same may not be the case for other areas (such as the favelas of Rio de Janeiro). Nevertheless, the result is promising, and as we enhance the model of the households, so we may generate privacy-sensitive data, it is necessary to keep testing the generated profiles against real data in a similar manner.

Experiment 2: Detecting Fraudulent Households. In the first experiment we tested whether the simulated profiles of honest households are similar to real profiles of honest households. Due to the lack of real data regarding fraudulent households we cannot perform the same test for them. However, we do know that the ML techniques can learn to classify load profiles from real data as either honest or fraudulent. We can test our simulated data by verifying that the simulated profiles allow for a similar classification.

For this, we use the WEKA package. We tested a number of different ML settings: an SVM (similar to Nagi et al. [15]), a random forest (similar to the optimum-path forest classifier used by Ramos et al. [18], albeit less sophisticated), naive Bayes, and a multi-layer perceptron classifier [6].

We generate ten datasets with the same settings: 400 households, 15% of which are fraudulent, and using a 50/50 split of the profiles, adding Gaussian

Table 2. Average precision, recall and F_2 -measure for four different ML methods over 10 simulated datasets

ML method	Precision	Recall	F_2 -measure
SVM	0.98	0.87	0.89
Random Forest	0.96	0.55	0.60
Naive Bayes	0.97	0.90	0.91
Multilayer Perceptron	0.90	0.91	0.90

noise with $\sigma = 0.5$. We then use the four different learning algorithms with 10-fold cross validation. The average precision, recall and F_2 -measure (calculated as described in Section 3.2) are in Table 2.

Other than the random forest method, all give very similar results, and in fact, quite significantly better than the results of either Nagi et al. or Ramos et al., who had a precision and recall around 0.8 on real data. This indicates that our dataset is actually "too easy" to accurately represent real profiles. By increasing σ we can make it harder to learn a correct classification. Simply increasing σ to 0.6 gives a precision and recall that is more in line with the results in the related work, at the cost of a decrease in realism of the honest profiles (per Table 1).

A remarkable side result of this exploratory experiment is that we did not expect naive Bayes, a significantly simpler learning method than using an SVM or a multilayer perceptron classifier, to perform so well. We have not seen naive Bayes applied on real data in any of the related work we studied, and it is worth investigating whether it gives similar results in practice.

5 Discussion and Future Work

Preservation of private information is a real concern for many modern applications of information technology. When deploying smart meters such privacy concerns should be addressed without compromising the benefits of using smart meters in the first place. The methodology we present in Section 3 makes explicit the trade-off between privacy-preservation and extraction of important information; in this case whether a household is fraudulent or not. Using a multi-agent system to model a neighbourhood allows for the flexible implementation of various different consumer profiles and their possible interactions.

The results of Section 4 validate our proof-of-concept implementation of the method, by comparing the simulated neighbourhood to real data. We argue that this demonstrates a novel method for understanding the social impact of smart sensor technology. The methodology as presented allows for analysing the impact of smart grid technology on different communities.

The next step is to generate household profiles that contain privacy-sensitive information, and generate the household's electricity load profiles based on these, while verifying that this is still realistic, in the same sense as our currently

simulated data. These load profiles can then be used to verify that privacy-sensitive data can be discovered through data mining techniques, and with this simulation in place we can move on to both testing current privacy-preservation methods and designing new ones that optimise the trade-off between privacy and knowledge discovery.

While we are not there yet, this paper presents a step towards a privacy-conscious use of data from smart meters.

Acknowledgements. Gabriel Ramos and Ana Bazzan are partially supported by CNPq, Andrew Koster is supported by CAPES (PNPD). All three are supported in their work by FAPERGS. This research was done in collaboration with IBM Research Brazil.

References

1. Aggarwal, C.C., Yu, P.S.: A general survey of privacy-preserving data mining models and algorithms. In: *Privacy-Preserving Data Mining. Advances in Database Systems*, vol. 34, pp. 11–52. Springer, Heidelberg (2008)
2. Agrawal, D., Aggarwal, C.C.: On the design and quantification of privacy preserving data mining algorithms. In: *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 247–255. ACM (2001)
3. Armstrong, M.M., Swinton, M.C., Ribberink, H., Beausoleil-Morrison, I., Jocelyn, M.: Synthetically derived profiles for representing occupant-driven electric loads in canadian housing. *Journal of Building Performance Simulation* 2(1), 15–30 (2009)
4. Baeza-Yates, R., Ribeiro-Neto, B.: *Modern Information Retrieval: The Concepts and Technology behind Search*, 2nd edn. ACM Press (2011)
5. Bertino, E., Lin, D., Jiang, W.: A survey of quantification of privacy preserving data mining algorithms. In: *Privacy-Preserving Data Mining. Advances in Database Systems*, vol. 34, pp. 183–205. Springer, Heidelberg (2008)
6. Bishop, C.M.: *Pattern Recognition and Machine Learning*. Springer (2006)
7. Buttarelli, G.: Opinion of the European data protection supervisor on the commission recommendation on preparations for the roll-out of smart metering systems. *EU Recommendation* (2012)
8. Cabral, J., Gontijo, E.: Fraud detection in electrical energy consumers using rough sets. In: *2004 IEEE International Conference on Systems, Man and Cybernetics*, pp. 3625–3629 (2004)
9. Cohen, F.: The smarter grid. *IEEE Security Privacy* 8(1), 60–63 (2010)
10. de Almeida, A., Fonseca, P., Bandeirinha, R., Fernandes, T., Araújo, R., Urbano, N., Dupret, M., Zimmermann, J.P., Schlomann, B., Gruber, E., Kofod, C., Feildberg, N., Grinden, B., Simeonov, K., Vorizek, T., Markogianis, G., Giakoymi, A., Lazar, I., Ticuta, C., Lima, P., Angioletti, R., Larssonneur, P., Dukhan, S., de Groote, W., de Smet, J., Vorsatz, D., Kiss, B., Ann-Claire, L., Pagliano, L., Roscetti, A., Valery, D.: *REMODECE: Residential monitoring to decrease energy use and carbon emissions in europe. Technical report, IIEA Programme* (2008)
11. Erkin, Z., Troncoso-Pastoriza, J.R., Legendijk, R.L., Pérez-González, F.: Privacy-preserving data aggregation in smart meter systems. *IEEE Signal Processing Magazine* 30(2), 75–86 (2013)

12. IBM Corp.: IBM Smart Grid, <http://www.ibm.com/smarterplanet/energy> (last checked May 2013)
13. Kadurek, P., Blom, J., Cobben, J.F.G., Kling, W.: Theft detection and smart metering practices and expectations in the netherlands. In: 2010 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT Europe), pp. 1–6 (2010)
14. Laughman, C., Lee, K., Cox, R., Shaw, S., Leeb, S., Norford, L., Armstrong, P.: Power signature analysis. *IEEE Power and Energy Magazine* 1(2), 56–63 (2003)
15. Nagi, J., Yap, K., Tiong, S.K., Ahmed, S., Mohamad, M.: Nontechnical loss detection for metered customers in power utility using support vector machines. *IEEE Transactions on Power Delivery* 25(2), 1162–1171 (2010)
16. Nagi, J.: An intelligent system for detection of non-technical losses in Tenaga Nasional Berhad (TNB) Malaysia low voltage distribution network. Master's thesis, University Tenaga Nasional (June 2009)
17. Paatero, J.V., Lund, P.D.: A model for generating household electricity load profiles. *International Journal of Energy Research* 30(5), 273–290 (2006)
18. Ramos, C.C.O., Souza, A.N., Papa, J.P., Falcão, A.X.: Learning to identify non-technical losses with optimum-path forest. In: Proceedings of the 17th International Conference on Systems, Signals and Image Processing (IWSSIP 2010), pp. 154–157 (2010)

Event-Driven Programming for Situated MAS with ReSpecT Tuple Centres

Stefano Mariani and Andrea Omicini

DISI, ALMA MATER STUDIORUM—Università di Bologna
via Sacchi 3, 47521 Cesena, Italy
{s.mariani, andrea.omicini}@unibo.it

Abstract. We advocate the role of tuple-based coordination languages as effective tools for event-driven programming of situated multi-agent systems (MAS). By focussing on logic-based coordination artefacts, we discuss the benefits of exploiting ReSpecT tuple centres as event-driven abstractions for MAS coordination.

1 Introduction

The role of the *environment* in multi-agent systems (MAS) is nowadays widely recognised: as a foremost source of complexity, environment should be handled as a first-class entity in MAS modelling and engineering [1]. This requires suitable abstractions, technologies, and methodologies, which are typically provided by *agent middleware* [2]—from JADE services [3] to CArtAgO artefacts [4].

The Agent & Artefact (A&A) meta-model adopts agents and *artefacts* as the basic bricks for MAS. There, artefacts are first-class abstractions to model and engineer general-purpose computational MAS environments [5]. In particular, *environment artefacts* (also called *resource artefacts*) are in charge to mediate between MAS and the resources in the virtual and physical environment [6]. Therefore, environment artefacts are the fundamental abstractions to deal with agent and MAS *situatedness*, as the property of being immersed in the environment, that is, capable to perceive and produce environment change, by suitably dealing with *environment events*.

Agent-oriented *event-driven programming* models and languages are then required to design and implement *situated MAS*—such as pervasive, adaptive, and self-organising MAS [7]. Accordingly, in this paper we focus on the TuCSon coordination middleware for MAS [8], and show how its *coordination artefacts* [9] – that is, ReSpecT tuple centres [10] – provide the required event-driven model and technology for programming situated MAS. To this end, Section 2 shortly discusses the notion of artefact in the A&A meta-model, the notion of coordination artefact, and its relation with tuple-based coordination models. Then, Section 3 describes ReSpecT tuple centres as providers of an event-driven model and technology for programming fully situated MAS. Finally, Section 4 discusses some examples of MAS event-driven programming in ReSpecT.

2 Artefacts, Coordination and Tuple Centres

Given the ever-growing relevance of MAS abstractions and technologies in the engineering of pervasive [11,12], adaptive [13,14], and self-organising systems [15,16] – where the environment works as the main source of system complexity –, the issue of MAS environment engineering is nowadays of paramount importance [1,2]. Accordingly, the property of individual agents and of the MAS as a whole to be *immersed* into the environment, perceiving any relevant environment change, and interacting effectively with environment resources – in one word, MAS *situatedness* – should definitely play a key role in MAS models and technologies. In particular, one of the main issues here is how agents and MAS are expected to properly deal with possibly unpredictable and unstructured *environment events* of any sort.

Along this line, the Agent & Artefact (A&A) meta-model adopts *artefacts* as the main abstraction for modelling and engineering general-purpose computational MAS environments [5]. As depicted in Fig. 1, in A&A *individual artefacts* rule interaction between individual agents and MAS, *social artefacts* embody and enact social policies to govern inter-agent interaction, *environment artefact* (also called *resource artefacts*) are in charge to mediate between MAS and the resources in the virtual and physical environment [6]. Then, environment artefacts are the fundamental abstractions to deal with MAS situatedness, in charge of suitably dealing with environment events: however, as a meta-model, A&A does not directly provide neither the required computational model nor the technology to build environment artefacts.

On the other hand, the notion of *coordination artefacts* aims at generally systematising implicit communication and environment-based coordination for heterogeneous, possibly intelligent agents in MAS, by providing MAS engineers with a class of abstractions capable of dealing with general-purpose *coordination events* in a MAS [9]. Among the many available sources for coordination artefacts, *tuple-based coordination models* already proved their effectiveness in coping with complex MAS engineering [17], mostly due to the following features:

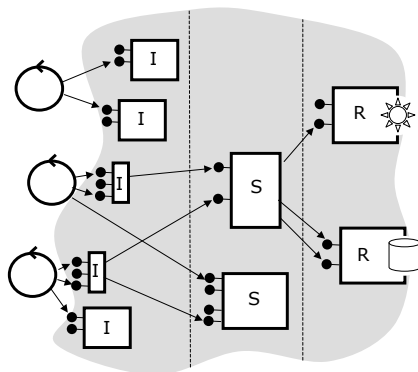


Fig. 1. Class of artefacts according to the A&A meta-model: I are *individual artefacts*, S are *social artefacts*, R are *environment*, (or, *resource*) artefacts [6]

Generative Communication — Information exchanged among agents, recorded in the *tuple space* in terms of *tuples*, has an independent life with respect to its “generator” agent, thus promoting *full communication uncoupling* (space, time, name) among agents. If tuples represent events, then events inherit such features, promoting full uncoupling from their source.

Associative Access — Information retrieval is based on tuple *structure* and *content*, rather than on name, or location, thanks to the tuple matching mechanism, thus allowing for *knowledge-based coordination patterns*. Again, when tuples reify events, then any event is available to any agent able to associatively access to it.

Suspensive Semantics — Agents block on tuple space operations if the information they look for is missing, thus enabling coordination policies in incomplete or partial knowledge scenarios. Also, reactive behaviours, which are desirable in event-based systems, are thus naturally supported by simply requesting the expected event-tuple and relying on LINDA semantics.

Nevertheless, the most relevant drawback of such models lies in their lack of *expressiveness*: possible coordination policies are determined by the available coordination primitives – e.g. *out*, *in*, *rd* in LINDA [18] –, along with the corresponding behaviour of the tuple space, set once and for all by the model [19]. Any coordination requirement not directly reducible to the available coordination primitives should then be faced by charging coordination upon the agents: which is acceptable for *individual* coordination issues, not for *social* ones [20].

This is the main motivation behind the definition of *tuple centres* as a *programmable* extension of tuple spaces [10]. Along with the *communication space*, represented by the tuples exchanged by agents, a tuple centre provides a *coordination space*, storing a *behaviour specification* written in terms of *specification tuples*, that sets the coordinative behaviour of the coordination media. Each tuple centre can be programmed so as to perform some computational activity in response to *events* involving the coordination medium. In this way, the behaviour of a tuple centre – and so, the MAS coordination policies it can handle – can be made as complex as needed by the application at hand. As a result, coordination issues in a MAS can be faced according to basic engineering principles: individual ones within individual abstractions (agents), social ones within social abstractions (tuple centres) [20].

Once re-casted as (coordination) artefacts [21], tuple centres exhibit the following artefact features:

Inspectability — The capability of artefact *state and behaviour* to be observed at runtime, thus supporting their on-line management in terms of diagnosing, debugging, testing. When tuple centres are *event mediators*, this actually means making any event occurring there *observable*.

Forgeability — The capability of artefact *function* to be adapted at runtime according to changing requirements or *unpredictable events* occurring in an open environment. This makes it possible to change the way in which events are produced, managed, and consumed according to the application needs.

Linkability — The capability of distinct artefacts to work together at runtime as a form of *dynamic composition*, e.g., to scale up with the complexity of the computational load. This is essential to make the architecture in Fig. 1 feasible, allowing any event to be arbitrarily and seamlessly spread within the MAS at hand, building *event chains*. Therefore, any MAS event has both a *prime cause* and a *direct cause*: the prime cause is always an agent or the environment, whereas the direct cause may also be another tuple centre [21].

Situatedness — The property of being immersed in the environment, perceiving and reacting to *environment events* and changes: essential in complex software systems such as pervasive, adaptive and self-organising ones [7].

In particular, situatedness is the tuple centre property required to address the issue of *agent-environment coordination* as the last fundamental MAS coordination issue, after individual and social ones [22]. As sketched in Fig. 1, the A&A meta-model promotes an event-driven view of MAS: thus, a MAS coordination model exploiting event-driven coordination media is apparently a fit solution for addressing the issue of MAS situatedness, suitably governing interaction between agents, MAS, and environment.

Along this line, in the following we focus on the TuCSon middleware for MAS coordination [8], and investigate how the features of ReSpecT tuple centres – TuCSon coordination artefacts – can be effectively exploited to build an event-driven model for MAS situated coordination, by exploiting ReSpecT as an event-driven programming language.

3 ReSpecT Tuple Centres for Event-Driven Coordination

3.1 ReSpecT Event Model

In ReSpecT, a general tuple centre *event* is either an *operation* event, an *environment* event, or a *spatio-temporal* event:

$$\langle Event \rangle ::= \langle OpEvent \rangle \mid \langle EnvEvent \rangle \mid \langle STEvent \rangle$$

Operation Events An *operation event* is any event whose *prime cause* is an agent (coordination) operation. Inspectability is exploited by making observable (i) the primitive invoked along with its argument, (ii) the “source” agent who issued it, and (iii) the tuple centre target of such operation:

$$\langle PrimeCause \rangle ::= \langle CoordOp \rangle, \langle AgentId \rangle, \langle TCId \rangle$$

Due to linkability, the *direct cause* can instead be either a coordination operation or a *linking primitive* invoked within a ReSpecT reaction—thus executed by another ReSpecT tuple centre:

$$\langle DirectCause \rangle ::= \langle CoordOp \mid LinkOp \rangle, \langle AgentId \mid TCId \rangle, \langle TCId \rangle$$

Finally, the completion of any operation invocation typically has some *effect* on the tuple centre, namely, the tuple inserted/collected. Hence, operation events have roughly the following structure in ReSpecT:

$$\langle OpEvent \rangle ::= \langle PrimeCause \rangle, \langle DirectCause \rangle, \langle Tuple \rangle$$

Environment Events. According to the A&A metamodel, *environment artefacts* are required to mediate between agents of a MAS and their environment. In TuCSon, this role is played by ReSpecT tuple centres through (*event*) *transducers* [22], that is, components able to bring environment-generated events within a tuple centre, suitably translated according to its event model. As a result, from the tuple centre viewpoint, a transducer is technically both the prime and direct cause of any environment event, working as a sort of proxy for an *environment resource*, which represents the actual source of the event.

So, in the very end, an *environmental event* for a tuple centre has basically the following structure (more on this in [22,23]):

$$\begin{aligned} \langle EnvEvent \rangle &::= \langle EnvCause \rangle, \langle EnvCause \rangle, \langle env(Key, Value) \rangle \\ \langle EnvCause \rangle &::= \langle \leftarrow \mid \rightarrow \rangle \langle env(Key, Value) \rangle, \langle EnvResId \rangle, \langle TCId \rangle \end{aligned}$$

Space-Time Events. By adopting the same approach of [22], events related to the flow of time or to motion in space could be interpreted as being generated by the MAS environment. Along this line, we could model them, too, as environmental events, leaving their management to properly designed transducers. However, this approach falls short in recognising a fundamental difference between an event generated by the environmental resources and one related to the space-time fabric: whereas any resource *may* or *may not* be represented as part of the environment of a MAS, time flows, and space may change, within *any* MAS. So, spatial events [23] and time events [24] both mandate for a first-class status in event-driven programming for MAS coordination.

On the one hand, the flow of time is something not an agent neither a coordination artefact can control, hence the only time event to consider is the ever-changing current time. On the other hand, the space where a MAS tuple centre is executing can be perceived, but also changed through appropriate motion actions. Furthermore, perception and action over the space fabric can be described at two different levels, at least: the *physical* and the *virtual* one. The physical level is that of the *physical world* we live in, made up of *physical places* and *physical distances* detected through GPS services. The physical level is, e.g., that of a swarm of robots who need to coordinate themselves to explore an unknown area as best as they can, hence transducers capable of both perceiving GPS/accelerometer data and of issuing motion commands to such devices may be available. The virtual level is that of the *virtual world* the digital era brought us, made up of *virtual places* and *virtual distances* defined by network services and protocols such as the TCP/IP for the Internet. The virtual level is, e.g., that of a swarm of software agents who need to coordinate themselves to collect hyperlinking information in a pool of web sites as fast as they can, hence they could need to relocate network resources as a consequence to their mobility.

Accordingly, a *space-time event* for a tuple centre has the following structure:

$$\begin{aligned} \langle STEvent \rangle &::= \langle STCause \rangle, \langle STCause \rangle, \langle SOP \mid TOP \rangle \\ \langle STCause \rangle &::= \langle SOP \mid TOP \rangle, \mathbf{time} \mid \mathbf{space}, \langle TCId \rangle \\ \langle SOP \rangle &::= \mathbf{from}(Place) \mid \mathbf{to}(Place) \mid \mathbf{node}(Node) \\ \langle TOP \rangle &::= \mathbf{time}(Time) \end{aligned}$$

where *Time* is the current time of the tuple centre execution environment, *Place* represent the current GPS coordinates – hence physical, absolute and global to all the networked tuple centres – and *Node* is some identifier of the tuple centre network node, making it reachable from remote devices—e.g. its IP address, or the identifier of the 3G cell.

Operation & Environmental Events in the Space-Time Fabric. If the tuple centre is spatio-temporally situated – that is, able to recognise events occurring in the space-time fabric, and to properly react –, then any event it generates, manages, and consumes should be spatio-temporally situated as well. Therefore, both operation events and environmental events need to be refined so as to add space-time fabric properties and spatio-temporal events should be properly completed, too. In particular, all “cause” specifications defined above should be extended by adding the following information:

$$\langle Time \rangle, \langle Place \rangle, \langle Node \rangle$$

Such information may be available or not depending on the computational device hosting the tuple centre: in the case it is not, a “null” value is stored (\perp).

3.2 Events in the ReSpecT Language

As both a first-order logic (FOL) and an event-driven language, ReSpecT has both a declarative and a procedural part. As a specification language, it allows *events* of any sort to be declaratively associated to computational activities called *reaction goals* (*reactions*, often, in short) by means of specific logic tuples of the form $\mathbf{reaction}(E, G, R)$, called *specification tuples*.

In short, given a tuple centre event e , a specification tuple $\mathbf{reaction}(E, G, R)$ associates a reaction $R\theta$ to event e if $\theta = mgu(E, e)$ – where mgu is the first-order logic *most general unifier* – and guard $G\theta$ holds. Formally, the semantics of ReSpecT is defined in terms of the two functions \mathcal{Z} and \mathcal{E} .

\mathcal{Z} (Reaction Specification Function). \mathcal{Z} defines how ReSpecT associates events to reactions. If e is an event occurring in a tuple centre whose behaviour specification σ (a ReSpecT program) contains reactions r of the form $\mathbf{reaction}(E, G, R)$, then the reaction goals to be executed are defined by \mathcal{Z} as follows:

$$\mathcal{Z}_\sigma(e) ::= \biguplus_{r \in \sigma} z(e, r), \quad \text{where } z(e, r) ::= \begin{cases} {}^eR\theta & \text{if } \theta = mgu(E, e) \wedge G\theta \\ \emptyset & \text{otherwise} \end{cases}$$

that is, collecting all reactions goals ${}^eR\theta$ whose triggering event E matches the specific event e modulo unification (θ), provided that guard $G\theta$ holds. Triggered reaction goals are then executed following the reaction evaluation function \mathcal{E} .

\mathcal{E} (Reaction Evaluation Function). \mathcal{E} encapsulates ReSpecT conceptual machinery for reaction execution, which is essentially an event-driven virtual machine collecting events and properly reacting according to a ReSpecT program.

In short – more details in [25,21,26,22,23] –, the state of a ReSpecT tuple centre is formally expressed by the labelled quadruple

$$InQ, EnvQ \langle \mathcal{T}, \sigma, \mathcal{R}, \mathcal{O} \rangle OutQ$$

where \mathcal{T} is the set of tuples, σ the set of specification tuples (the behaviour specification), \mathcal{R} the set of triggered reaction goals to be executed, \mathcal{O} the set of operations waiting to be served, whereas InQ , $EnvQ$, $OutQ$ are event queues containing input and output events. In particular, InQ contains *OpEvents* representing coordination primitives invoked either by agents (operations) or tuple centres (links); $EnvQ$ contains both *EnvEvents* and *STEvents* – so, environment events in the most general acceptance of the term –; $OutQ$ contains all kinds of output events generated by reaction execution within a tuple centre.

Basically, the main cycle of a tuple centre work as follows—formally, four transition rules drive ReSpecT machinery, as originally described in [26] where the only environment event modelled was time:

1. When a tuple centre primitive is invoked by either an agent (operation) or an artefact (link), a ReSpecT event is generated, and reaches its target tuple centre, where it is automatically and orderly inserted in its InQ queue.
2. When the tuple centre is idle (that is, no reaction is currently being executed), the first event in InQ (according to a FIFO policy) is logged, and moved to the set \mathcal{O} of the requests to be served: this stage is called the *request phase* of the event. This corresponds to the *log* transition rule, firing when $\mathcal{R} = EnvQ = \emptyset$, \mathcal{O} contains no satisfiable requests, and $InQ \neq \emptyset$.
3. Consequently, reactions to the request phase of the event are triggered by adding them to the set \mathcal{R} of the triggered reactions waiting to be executed.
4. All triggered reactions in \mathcal{R} are then executed in a non-deterministic order. Each reaction is executed sequentially, with a transactional semantics, and may trigger further reactions, to be added to \mathcal{R} , and new output events, representing link invocations: such events are added to the queue $OutQ$ of output events, and released at the end of reaction execution—if successful. This phase is driven by the *reaction* transition rule, firing when $\mathcal{R} \neq \emptyset$.
5. Only when \mathcal{R} is finally empty, requests waiting to be served in \mathcal{O} are possibly executed by the tuple centre, and operation/link completions are sent back to invokers. The corresponding transition rule is the *service* transition, firing when $\mathcal{R} = EnvQ = \emptyset$ and \mathcal{O} contains at least one satisfiable request.
6. This may give raise to further reactions, associated to the response phase (completion) of the original invocation, and executed again with the same semantics specified above for the request phase.

Whereas InQ is fed by coordination primitives, $EnvQ$ is added new environment events by the associated *transducers* translating environment events into ReSpecT events, as well as by the flow of time, and by motion in space [26,22]. The transition rule responsible of this is the *environment* transition, firing when $\mathcal{R} = \emptyset$ and $EnvQ \neq \emptyset$.

Since the full definition of the ReSpecT syntax and semantics falls out of the scope of this paper, we forward the interested reader to [25,21,26,22,23].

As a last note, it is indeed useful to highlight once more a crucial feature of the ReSpecT virtual machine: *decoupling in control*. Whereas TuCSoN agents can autonomously decide whether to suspend themselves waiting for the outcome of an issued operation – following LINDA suspensive semantics –, ReSpecT tuple centres linkability service is always *asynchronous*. This means that the *invocation phase* (point 1 in previous list) and the *completion phase* (point 5) are always handled asynchronously by ReSpecT virtual machine:

- once linking invocations are sent out – put in the target tuple centre *InQ* –, ReSpecT tuple centre machinery continues onward to step 3, without waiting for the linking completion;
- when linking completions have to be sent back by the target tuple centre, they are simply inserted into the *InQ* of the source tuple centre, and processed asynchronously.

This kind of uncoupling enables scaling with the size and complexity of the MAS at hand, because the programmer is guaranteed by the ReSpecT language that neither interference or deadlock can happen between linking operations.

4 Event-Driven Programming in ReSpecT: Examples

In the following, we discuss some examples of MAS event-driven programming in ReSpecT, with the twofold goal of showing: *(i)* how ReSpecT logic tuple centres can be programmed – exploiting their A&A features – by mapping events into computations; *(ii)* how ReSpecT event model and language constructs can help the process of developing a situated MAS.

4.1 Generative Communication, Associative Access and Inspectability

Generative communication, *associative access*, and *inspectability* features exhibited by ReSpecT tuple centres can be easily highlighted through the following ReSpecT specification snippet.

```

1 reaction( in(Tuple),
2   ( operation, invocation ),
3   ( start_source(Src),           % Prime Cause inspection
4     start_target(Trg),
5     event_time(T),              % Direct Cause inspection
6     event_node(N),
7     out( event(req(in(Tuple)),from(Src),to(Trg),at(T),on(N)) ) % Reification
8 ) ).

```

The above ReSpecT specification transparently – from the agents' standpoint – encapsulates *situated inspection & reification* of an operation request (*invocation* guard) performed by an agent (*operation*). In fact, the operation request (in the example, *in(Tuple)*) is augmented with contextual information taken from the generated ReSpecT event – *who* was the original requestor (observation predicate *start_source*) and the original destination (*start_target*) of the operation,

at what time (`event_time`) and on which host (`event_node`) such event was generated – without no need for the agent to be aware of it. Then, the situated event is stored in the tuple centre (line 7), made available for inspection and usage to anyone, being decoupled in time, space, and reference from its prime cause.

The integration of inspection and reification – and, obviously, forgeability – enables *event-based situated coordination* in an open environment: for instance, an agent may become aware of the existence of other agents (“*social awareness*”) by collecting the `Src` info, as well as of the availability of other coordination abstractions by recording the `Trg` info (“*environment awareness*”).

4.2 Logic and Associative Access

The following ReSpecT specification snippet is meant to highlight the gain in expressiveness given by *logic tuples* over other possible kind of tuples.

```

1 reaction( out(Tuple),
2   ( operation, completion ),
3   ( start_source(Src), start_target(Trg),
4     ( causalCheck( event(out(Tuple),from(Src),to(Trg)) ) % Prolog invocation
5       ; % 'if-then-else'
6         in(Tuple) % Resolve inconsistency
7     ) ) ).
8 causalCheck( event(Op, F, T) ):-
9   findall( B, clause(Op, B), L ), % Consult theory
10  causalCheck( F, T, L ).
11 causalCheck( _, _, [] ).
12 causalCheck( F, T, [H|Tail] ):-
13   rd(event(H, F, T)), % Check consistency
14   causalCheck( F, T, Tail ).

```

Here, we suppose to have a MAS where logical consistency must be enforced between (inter)actions performed by agents—e.g., insertion of some information is allowed only if other “causal” information exists. So, the goal of the ReSpecT specification above – along with the auxiliary Prolog predicates (lines 8-14) – is to verify such kind of “logical implication”, allowing insertion of a tuple if and only if all “precondition tuples” have been already stored. Knowledge about causal relationships is shipped as a logic theory – here, a Prolog one, encapsulated in the tuple centre, thus seamlessly integrated with ReSpecT – in which the head of a clause is a ReSpecT operation and its body a single precondition to be checked. If and only if all logic preconditions (`findall/3`) are actually found in the tuple centre, the insertion operation triggering the ReSpecT computation is allowed.

The snippet works as follows. Once an `out(Tuple)` operation has been completed (lines 1-2), the prime cause and the target of such event are inspected (line 3) to be used in the subsequent Prolog predicate invocation (line 4). Then, if Prolog evaluation succeeds, nothing is done, whereas if such evaluation fails, the inserted tuple is removed (line 6)—atomically deleting the *effect* of the triggering event, source of the inconsistency. The Prolog predicate responsible for consistency check (`causalCheck/1`) works as follows: it (*i*) collects all the preconditions to be checked, by retrieving all the clauses whose head matches the ReSpecT operation causing the event (line 9), then (*ii*) tests non-destructively

the existence of each precondition (only the body B of the clause is collected) in the tuple centre (lines 12-14).

The logical nature of ReSpecT tuples – along with inspectability and forgeability, as usual – enables “*intelligent*” agents to perform logical reasoning upon coordination events, leveraging a simple data store (a tuple space) to a logic theory about coordination events (a ReSpecT tuple centre). In fact, with just a slight modification to the ReSpecT specification sketched above – e.g., by reifying Prolog clauses as tuples –, an agent may even *learn* at run-time which kind of coordination operations it is supposed to carry out before others, by performing *inference* on the logical (event-driven) coordination theory.

4.3 Situatedness and Linkability

The following ReSpecT code snippet sketches a ReSpecT program meant to coordinate the motion of robot swarms so as to reach a “uniform” distribution in space, in which each robot is equidistant from its neighbours—modulo small fluctuations due to the *local* and *distributed* nature of the proposed solution.

```

1  reaction( out(moved(NewPos)),
2    ( link_in, completion ),           % Incoming link
3    ( in(moved(NewPos)),
4      event_source(Who),               % Direct cause inspection
5      in(nbr(Who, Pos)), out(nbr(Who, NewPos)), rd_all(_, _, Nbrs),
6      computeBaricenter(Nbrs, B),      % Prolog invocation
7      current_node(Host),             % Inspect actual host
8      steering_engine@Host <- env('destination', B), % Command actuator 1
9      motion_engine@Host <- env('power', 'on')      % Command actuator 2
10 ) ).
11 reaction( from(Pos),                 % 'Motion awareness'
12   ( internal ),
13   ( % Avoid obstacles, collisions, etc.
14     % Monitor arrival to destination
15 ) ).
16 reaction( to(NewPos),                % 'Stillness awareness'
17   ( internal ),
18   ( rd_all(nbr(_, _), Nbrs), multicast(moved(NewPos), Nbrs)
19 ) ).
20 multicast( _, [] ).
21 multicast( Info, [nbr(N, _) | Nbrs] ) :-
22   N ? out(moved(Info)),               % Outgoing link
23   multicast(Info, Nbrs).

```

The first reaction (lines 1-10) is responsible for properly reacting to incoming *linking events* caused by the motion of neighbours: (*i*) after inspecting its cause (line 4, prime and direct causes coincide for linking operations), (*ii*) the position of the neighbours is updated (line 5), (*iii*) the new “center of gravity” among neighbours is computed (`computeBaricenter/2`), (*iv*) then host’s actuators are told what to do by setting their *environmental properties* (lines 8-9).

This reaction clearly shows the *context awareness* a ReSpecT tuple centre can provide: prior to setting actuators’ properties, the tuple centre inspects its own environment (line 7) to dynamically acquire knowledge about its execution context. In this way, even when some properties of the execution environment change (e.g. host’s network address) the reaction still works thanks to its situated nature. On the other hand, the *transparency* given by using logical names for ac-

tuators (`x_engine`) ensures the required degree of situatedness—since hardware is not expected to change often and dynamically.

The second reaction (line 11-15) is not shown completely for the lack of space: however, its purpose is to monitor both motion correctness and safety, as well as arrival to destination. Here, it is worth noting that the triggering event (`from(Pos)`), generated as a consequence of ReSpecT tuple centres *spatial awareness* enabling motion perception, promotes situatedness in space.

Last reaction (lines 16-19) is dual to the previous one: its triggering event is a change in “spatial state” from motion to stillness (`to(NewPos)`), generated as a consequence to actuators shutdown when destination is reached (done as part of reaction 11-15). When such an event occurs, new position is multicasted to neighbours robots (`multicast/2` predicate), ultimately making them trigger the whole reactions chain within their own tuple centre.

The goal of the ReSpecT code snippet above is to highlight not just the expressive power of events *situatedness* and *linkability* offered by ReSpecT tuple centres, but also two important features at the *software engineering* level:

Encapsulation — Each of the three reactions shown above is responsible for a specific sub-task in which the global task of achieving some spatial pattern (in this case, uniform distribution) can be decomposed: (i) motion start (lines 1-10), (ii) motion monitoring (lines 11-15), (iii) motion stop (lines 16-19).

Separation of concerns — The ability of (physical) agents to move into a (physical) environment demands for environment modelling and situatedness; modelling, however, should be carried out at the most suitable level of abstraction—e.g., an agent modelling an optical sensor might be just too much. The event-driven programming model enforced by ReSpecT provides exactly this kind of separation of concerns: agents should just care of their tasks (e.g., motion planning, motion policies adjustments, etc.) whereas environment modelling is left to environment abstractions such as tuple centres (e.g., generating motion events, reacting to sensors perceptions, etc.).

5 Related Works

The A&A metamodel [5] brings the notion of *artefact* to the MAS field. There, the admissible interactions between agents and artefacts – that is, *use* and *manifestation* – are clearly described. Accordingly, in [27] CArTAgO is proposed as a Java-based framework to design and engineer MAS based on A&A, providing agents a way to use artefacts – by executing operations –, and artefacts a way to manifest themselves to agents—by linking to sensors. Since the computational model adopted by CArTAgO artefacts is that of *passive* entities, merely reacting to agents stimuli, also the computational model of the MAS environment in CArTAgO is that of a passive context, in which nothing happens if not somehow caused by agents. Here instead, we foster the idea of a *proactive environment* able to autonomously generate events, which can then be perceived by interested agents upon need. Nevertheless, once a representation of such events is stored in

a tuple centre, and made available to agents, agents autonomy is preserved also *during* the interaction with artefacts, occurring through a shared environment.

A development environment for the engineering of self-adaptive MAS is proposed in [28]. Recognising the central role played by the environment in MAS, the *event* abstraction is adopted to design a self-adaptation mechanism based on the *organisation* metaphor. In particular, environmental events are notified to agents according to a publish/subscribe architecture, possibly triggering a change of role. Since agents' behaviours are bound to roles, switching roles leads to adaptation of behaviours. Although the role of events as the “glue” binding agents to their environment seems well understood, [28] does not exploit the environment abstraction to its full extent. In fact, environmental events are generated by other agents, playing the role of the part of the environment made observable to the MAS. Furthermore, situatedness of events is not clearly represented, since the only fields describing an event in [28] are: (i) the *type* of the event, which can describe a change in agents' state, behaviour, role or offered services; (ii) the *source* of the event (the agent who generated it); and (iii) a set of *constraints* whose function is not clearly explained.

In [29], the event abstraction is adopted to design a lightweight agent model called ELDA. In particular, any ELDA agent is a single-threaded autonomous entity interacting through asynchronous events, whose behaviour is expressed reactively in response to incoming events. Although ELDA does not natively support any environment abstraction, it has been extended to support the PACO model abstractions [30], splitting MAS into four parts: agents, environment, interactions, and organisation. Nevertheless, such an extension seemingly accounts for agents' position only, and the only agent-environment interaction is due to a *monitor agent* which continuously monitor the environment and the agents' state, triggering events when some pre-conditions are met.

6 Conclusions

In this paper, we show how a model for MAS coordination can be used to re-cast a MAS as an event-driven system, and consequently how a coordination language can be exploited for event-driven programming of situated MAS. In particular, we discuss the main benefits of tuple-based coordination models for event-driven programming, by focussing on the ReSpecT logic-based tuple centres provided by the TuCSoN coordination middleware, and elaborating on the notions of environment and spatio-temporal events from a tuple centre perspective. After a brief overview of the ReSpecT event-driven coordination framework, we present a few examples to showcase ReSpecT expressive power when dealing with issues such as event situatedness in space and time.

Since a complete and stable implementation of ReSpecT situatedness predicates strictly depends on the underlying platform – e.g. the ability to get GPS data –, a reliable porting of TuCSoN and ReSpecT onto mobile platforms is currently ongoing. Current and future work is devoted to enhance “Mobile-TuCSoN” architecture and implementation so as to better tackle typical issues of mobile computing scenarios, e.g. transient connections, low memory, power constraints.

Acknowledgments. This work has been partially supported by the EU-FP7-FET Proactive project SAPERE – Self-aware Pervasive Service Ecosystems, under contract no. 256874.

References

1. Weyns, D., Omicini, A., Odell, J.J.: Environment as a first-class abstraction in multi-agent systems. *Autonomous Agents and Multi-Agent Systems* 14(1), 5–30 (2007)
2. Viroli, M., Holvoet, T., Ricci, A., Schelfhout, K., Zambonelli, F.: Infrastructures for the environment of multiagent systems. *Autonomous Agents and Multi-Agent Systems* 14(1), 49–60 (2007)
3. Bellifemine, F.L., Caire, G., Greenwood, D.: *Developing Multi-Agent Systems with JADE*. Wiley (February 2007)
4. Ricci, A., Viroli, M., Omicini, A.: *Construenda est CArTAgO: Toward an infrastructure for artifacts in MAS*. In: Trappl, R. (ed.) *Cybernetics and Systems 2006*, Vienna, Austria, April 18–21, vol. 2, pp. 569–574. Austrian Society for Cybernetic Studies (2006)
5. Omicini, A., Ricci, A., Viroli, M.: Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems* 17(3), 432–456 (2008)
6. Omicini, A., Ricci, A., Viroli, M.: *Agens Faber: Toward a theory of artefacts for MAS*. *Electronic Notes in Theoretical Computer Science* 150(3), 21–36 (2006)
7. Zambonelli, F., Castelli, G., Ferrari, L., Mamei, M., Rosi, A., Di Marzo Serungendo, G., Risoldi, M., Tchao, A.E., Dobson, S., Stevenson, G., Ye, Y., Nardini, E., Omicini, A., Montagna, S., Viroli, M., Ferscha, A., Maschek, S., Wally, B.: Self-aware pervasive service ecosystems. *Procedia Computer Science* 7, 197–199 (2011)
8. Omicini, A., Zambonelli, F.: Coordination for Internet application development. *Autonomous Agents and Multi-Agent Systems* 2(3), 251–269 (1999)
9. Omicini, A., Ricci, A., Viroli, M., Castelfranchi, C., Tummolini, L.: Coordination artifacts: Environment-based coordination for intelligent agents. In: Jennings, N.R., Sierra, C., Sonenberg, L., Tambe, M. (eds.) *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, vol. 1, pp. 286–293. ACM, New York (2004)
10. Omicini, A., Denti, E.: From tuple spaces to tuple centres. *Science of Computer Programming* 41(3), 277–294 (2001)
11. Mamei, M., Zambonelli, F.: *Field-Based Coordination for Pervasive Multiagent Systems. Models, Technologies, and Applications*. Springer Series in Agent Technology. Springer (March 2006)
12. Omicini, A., Ricci, A., Vizzari, G.: Building smart environments as agent workspaces. In: *IEEE 16th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2007)*, 1st International Workshop on Interdisciplinary Aspects of Coordination Applied to Pervasive Environments: Models and Applications (CoMA 2007), June 18–19, pp. 92–97. IEEE CS, Paris (2007)
13. Guessoum, Z.: Adaptive agents and multiagent systems. *IEEE Distributed Systems Online* 5(7) (July 2004)
14. Cernuzzi, L., Molesini, A., Omicini, A., Zambonelli, F.: Adaptable multi-agent systems: The case of the Gaia methodology. *International Journal of Software Engineering and Knowledge Engineering* 21(4), 491–521 (2011)

15. Di Marzo Serugendo, G., Gleizes, M.P., Karageorgos, A.: Self-organization in multi-agent systems. *The Knowledge Engineering Review* 20(2), 165–189 (2005)
16. Gardelli, L., Viroli, M., Omicini, A.: Design patterns for self-organising systems. In: Burkhard, H.-D., Lindemann, G., Verbrugge, R., Varga, L.Z. (eds.) *CEEMAS 2007. LNCS (LNAI)*, vol. 4696, pp. 123–132. Springer, Heidelberg (2007)
17. Omicini, A., Viroli, M.: Coordination models and languages: From parallel computing to self-organisation. *The Knowledge Engineering Review* 26(1), 53–59 (2011)
18. Gelernter, D.: Generative communication in Linda. *ACM Transactions on Programming Languages and Systems* 7(1), 80–112 (1985)
19. Denti, E., Natali, A., Omicini, A.: Programmable coordination media. In: Garlan, D., Le Métayer, D. (eds.) *COORDINATION 1997. LNCS*, vol. 1282, pp. 274–288. Springer, Heidelberg (1997)
20. Ciancarini, P., Omicini, A., Zambonelli, F.: Multiagent system engineering: The coordination viewpoint. In: Jennings, N.R. (ed.) *ATAL 1999. LNCS (LNAI)*, vol. 1757, pp. 250–259. Springer, Heidelberg (2000)
21. Omicini, A.: Formal ReSpecT in the A&A perspective. *Electronic Notes in Theoretical Computer Science* 175(2), 97–117 (2007)
22. Casadei, M., Omicini, A.: Situated tuple centres in ReSpecT. In: Shin, S.Y., Ossowski, S., Menezes, R., Viroli, M. (eds.) *SAC 2009, March 8-12, vol. III*, pp. 1361–1368. ACM, Honolulu (2009)
23. Mariani, S., Omicini, A.: Promoting space-aware coordination: ReSpecT as a spatial-computing virtual machine. In: *Spatial Computing Workshop (SCW 2013), AAMAS 2013, Saint Paul, Minnesota, USA (May 2013)*
24. Omicini, A., Ricci, A., Viroli, M.: Timed environment for Web agents. *Web Intelligence and Agent Systems* 5(2), 161–175 (2007)
25. Omicini, A., Denti, E.: Formal ReSpecT. *Electronic Notes in Theoretical Computer Science* 48, 179–196 (2001)
26. Omicini, A., Ricci, A., Viroli, M.: Time-aware coordination in ReSpecT. In: Jacquet, J.-M., Picco, G.P. (eds.) *COORDINATION 2005. LNCS*, vol. 3454, pp. 268–282. Springer, Heidelberg (2005)
27. Ricci, A., Viroli, M., Omicini, A.: CArtAgO: A framework for prototyping artifact-based environments in MAS. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) *E4MAS 2006. LNCS (LNAI)*, vol. 4389, pp. 67–86. Springer, Heidelberg (2007)
28. Dong, M., Mao, X., Yin, J., Chang, Z., Qi, Z.: Sade: A development environment for adaptive multi-agent systems. In: Yang, J.-J., Yokoo, M., Ito, T., Jin, Z., Scerri, P. (eds.) *PRIMA 2009. LNCS*, vol. 5925, pp. 516–524. Springer, Heidelberg (2009)
29. Fortino, G., Garro, A., Mascillaro, S., Russo, W.: Using event-driven lightweight DSC-based agents for MAS modelling. *International Journal of Agent-Oriented Software Engineering* 4(2), 113–140 (2010)
30. Hallenborg, K., Jensen, A.J., Demazeau, Y.: Reactive agent mechanisms for manufacturing process control. In: *2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology Workshops (WI-IATW 2007)*, pp. 399–403. IEEE Computer Society, Washington, DC (2007)

HySoN: A Distributed Agent-Based Protocol for Group Formation in Online Social Networks

Fabrizio Messina¹, Giuseppe Pappalardo¹, Domenico Rosaci^{2,3}, Corrado Santoro¹,
and Giuseppe M.L. Sarné²

¹ Dipartimento di Matematica e Informatica, Università di Catania,
V.le Andrea Doria 6, 95125 Catania, Italy

{messina, pappalardo, santoro}@dmi.unict.it

^{2,3} DIIES University “Mediterranea” of Reggio Calabria,

DICEAM University “Mediterranea” of Reggio Calabria,

via Graziella, Feo di Vito 89122 Reggio Calabria (RC), Italy

{domenico.rosaci, sarné}@unirc.it

Abstract. On-line social networks allow people to easily interact with each other by means of social computer services. This scenario makes possible to search in a social network for affinities or new opportunities that satisfy specific requirements. However, for many users such activities often imply undesirable accesses to personal sensitive data. In this scenario we propose a novel approach, called HySoN (Hyperspace Social Network), based on an overlay network of software agents. HySoN allows users to locally maintain sensitive user’s data, satisfying the privacy requirements preserving sensitive data. Indeed, the properties involved in the HySoN user aggregation are inferred by local data not published in the social network. Some experimental results obtained on simulated on-line social networks data show the searching of suitable nodes is very efficient due to the topology of the overlay network, which exhibits the small-world properties.

Keywords: Software agent, overlay network, social network, gossip protocol, group formation.

1 Introduction

The rapid diffusion of general purpose social networks like Facebook¹, MySpace², Twitter³, and thematic networks as, for instance, Flickr⁴ or LinkedIn⁵, is representing an important opportunity for people to socialise, share multimedia contents, discover work opportunities and so on. Moreover, these online social environments are also exploited by companies to attract new clients and partners. In such a context users would like to find interlocutors with good affinity, but often this affinity can be found only by analysing sensitive data that they don’t want to share, therefore important privacy concerns arise for users.

¹ www.facebook.com

² www.myspace.com

³ www.twitter.com

⁴ www.flickr.com

⁵ www.linkedin.com

In this work we propose an approach for supporting the users of an online social network to form homogeneous environments (groups), suitable for performing social activities as sharing multimedia contents, making discussions, and so on. This approach is based on an overlay network of software agents [16], which allows the users to locally maintain sensitive user's data, while contacting interlocutors having a certain level of affinity is still made possible. The role of software agents consists in assisting users by analysing sensitive (local) data in order to extract relevant properties which can help in the social network to find other users having similar interests. To this aim, each agent can be delegated by its own user to start the construction of a custom overlay network called *HySoN* (Hyperspace Social Network) whose topology reflects the distribution of a set of users' interests (properties). In order to build such a network the software agents, totally trusted, exchange messages by a gossip protocol over the network of the friends. This way, as shown in the experimental section, approximately all the agents/nodes are reached by the overlay construction request in a negligible number of time-steps and an acceptable number of messages. Once the properties are mapped into the overlay network, the agents can send their requests over the network in order to find a set of agents reflecting some required properties. Moreover, as we will discuss in the experimental section, the searching of suitable nodes is very efficient due to the topology of the network, which exhibits the small-world properties. Finally, since properties involved in user aggregation are inferred by local data not published in the social network, the user privacy on sensitive data is basically preserved.

The remaining of the paper is organised as follows. In Section 2 we discuss related works. Section 3 describes the details of the basic scenario, while 4 introduces the technique for building the *HySoN* network and finding suitable nodes. Finally, an experimental evaluation of the proposed approach is provided in Section 5 and in Section 6 we draw some conclusions and future works.

2 Related Works

In the scientific literature the concept of social network has been widely investigated [11] and different definitions have been provided [47,49]. Generally, a social network can be described as the sum of the relationships connecting some individuals with each other [32], where a relationship embeds both personal knowledge and social resources [10].

This background is shared from on-line social networks [12,24], which are a worldwide phenomenon with a massive economic impact. Online social networks mainly differ from other social structures for the *(i)* the presence of a communication network, *(ii)* the absence of physical contacts and the *(iii)* potential presence of fake identities. In this way people interact with each other by using social computer services, giving rise new behavioural dynamics, widely studied in the recent years. Therefore, in this section, the examined approaches are those that, to the best of our knowledge, are the closest to the material presented in this paper, that deals with the formation of groups in online social networks.

Online social networks allow relationships and virtual groups to be created in a very simple and interactive way [27]. Virtual groups are formed, often in unplanned manner, on specific focuses, motivations or requirements and this process is often described

in the form of a Constraint Satisfaction Problem (CSP) [9]. The group membership is generally barrier free even though norms and requirements can exist [40], as well as playing well-defined roles statically or dynamically assigned (generally based on members' skills).

Different approaches are available for exploring the mechanisms underlying the group formation [39], although it lacks a well defined standard methodology. For instance, some analytic and synthetic measures (representing properties as cohesion, density, distances, etc. [45,47]) and the graph theory (to obtain a visual representation of the social network by modelling actors and relationships as nodes and arcs of a graph) can be used. Besides, the graph theory can be adopted to search potential candidates to form groups, by means of a merit function which involves into the creation of several partitions of the network [7,44].

Community members can be equipped with some kind of personal profile, potentially available to friends or similar entities. Such profiles store users' orientations and can be exploited for discovering potentially interesting groups. These processes can be driven also by means of machine-to-machine approach, for example by using the multi-agent technology [8,25,29] that offer several opportunities of performing them in a personalised and dynamic way.

In particular, a multi-agent system is naively implementable in a decentralised manner, similarly to HySoN. In this context, [20] exploits agents able to be adaptive with respect to changes occurring in the network topology by locally rewiring all their relationships and adapting themselves to the different social context. Also in [41], software agents are associated with users in order to manage dynamically and autonomously the evolution of the groups using by a distributed approach. Each software agent detects the most suitable groups to join with relying on a dissimilarity measure, while the agent associated with the group-administrator accepts only those join requests that come from users profile-compatible with the profile of the group.

Clustering algorithms [6,18,28,44] are a common way to discovery groups based on quantitative measures, usually extracted by data stored in personal profiles. They are classified in: (i) hierarchical [23] (recursively finding nested clusters either in agglomerative or divisive mode) and (ii) partitional (finding all the clusters simultaneously as a data partition without to impose a hierarchical structure). As a result, users are representable as points in a multidimensional space (the number of dimensions corresponds with the number of considered parameters in their profiles), where the Euclidean distance summarises the adopted clustering measure similarly to HySoN, although, as explained in the sections below, it works in a full distributed manner, hence it allows users not to maintain sensitive data in the remote server.

Two relevant issues might be taken into account in composing groups for specific tasks, i.e. (i) the workload balancing and (ii) the coordination costs. For instance, in [19,30,31] the first issue seems to be completely ignored, while in [1] the coordination costs are not considered, although in [2] both the workload distribution and the coordination costs are addressed. Moreover, in [31] the multiplicity of each group is a priori fixed by exploiting a modified version of the Enhanced-Steiner algorithm [30]. As we discuss in the Sections above, with our approach each member performs a little slice of the overall task to constitute a group; in such a way the first issue is addressed, while

the second issue is trivially solved by the approach adopted in the construction of the HySoN overlay network (see Section 4).

The dynamics of group formation have been recently addressed in [4], in the recent models proposed by Sarkar and Moore [43] and those discussed by Holme and Newman [22] to consider frequent changes occurring in the relationships and attributes. Furthermore an unsupervised discovery of groups in networks densely connected, similarly to many real-world networks, is proposed in [46]. Similarly, our proposal can automatically rearranging groups to take into account changes occurred in the social network even in presence of huge networks.

Greedy strategies are typical operational research techniques that can be applied, likewise to *HySoN*, in a social network scenario. For instance, they have been used on the GitHub network⁶ a Facebooklike platform, in order to allow the teams to self-organise and to solve specific tasks or minimise the team cost formation. Another greedy algorithm to solve the problem of realising a team of experts is described in [13], by which a weighted bipartite graphs with experts and another one with tasks are built in order to assign an expert to each task until experts or tasks end.

Finally, concerning fully decentralised finding algorithm, an interesting approach is presented in [14], where a two-layered gossip based protocol is exploited to deal with nodes failures, joins and leaves, leading to the construction of an additional random overlay network with a high level of redundancy. As we discuss in Section 4.2, our solution performs overlay maintenance by preserving the “essentially critical neighbours” within the overlay construction and maintenance algorithm with minimum cost, and do not lead to the construction of an additional network. Indeed, as explained in Section 4.2 and 4.3, the resulting network exploits the small-world properties, hence finding suitable nodes for group formation is possible with high efficiency.

3 Scenario

We consider a Social Network \mathcal{S} , having n users, and represented by a graph $G = \langle N, A \rangle$, where N is a set of nodes, such that each node $n \in N$ represents a user of \mathcal{S} that we denote by n_u , and A is a set of arcs, such that each arc $a = \langle u, v \rangle$ represents the existence of a friendship relation between the users n_u and n_v .

We assume that a software agent is allowed to reside on each user machine. This assumption partially overturns the architecture and the vision of the typical social network, on which user data reside on the Cloud [3,17,21] and the interactions between users occur through communications with the remote server. Nevertheless, as stated in Section 1, local software agents, which are trusted by users [42] are allowed to retrieve and analyse their personal (local) data which are considered “sensitive” by the users. Hence software agents can exploit a detailed profile based not only on data shared in the social network, but also on private data (e.g. emails). Hence the user profile can be used to create new groups of users with a decentralised technique (see Section 4).

From hereafter, we will use the terms *user* and *node* interchangeably. We also assume that a set P of *properties* is associated with the network of user, where a property can represent, for instance, the interest of a user for a set of topics, or the set of social

⁶ www.github.com

network pages that the node likes, or the set of social network groups which the node joins with, or the number of black and white photos which reside on the local disk of the user, and so on. Moreover, we also assume that each property $p \in P$ has a value belonging to a given domain D_p . For instance, if p represents the number of black and white photos held by the user and detected by the agent, then D_p will be a set of positive integer values, while if p represents the set of the topics which the agent consider relevant for the user, then D_p will be the set of all the possible sets of topics available in \mathcal{S} . A *profile schema* PS is also associated with the social network, composed of a list of properties belonging to P , denoted by p^1, p^2, \dots, p^l , where l is a value depending on the given social network. In this scenario, a *profile instance* p_n is associated with each node n of N , where p_n is a list $p_n^1, p_n^2, \dots, p_n^l$, such that each element p_n^i is an instance of the property $p^i \in PS$.

Each node, in order to evaluate if joining or not in a group with another node, can express some requirements on the properties of the other node. A *requirement on a property* p is a boolean function that accepts as input a property $p \in P$ and returns a boolean value $b \in \{true, false\}$. For instance, a requirement on the number of friends can state that a node must satisfy a given condition regarding the number of friends (e.g. having a number of friends higher than 50), returning the value *true* if the statement is verified by the property instance of the node, *false* otherwise.

Furthermore, each node n has associated a set of requirements R_n , and n accepts to belong to a group if at least the α percent of the members of the group satisfy at least the β percent of the requirements of R_n , where α and β are parameters set by the node n . Moreover, each node n can express his preferences regarding the group types he wants to join with, using the parameters *availability*, *accessibility* and *interests*, where *availability* is a boolean value, *accessibility* is a value belonging to the set $\{public, private, secret\}$ and *interests* is a set of topics. The value *availability* = *false* means that n does not desire to join with any group, while *availability* = *true* means that n is available to join with a new group under the condition that both the topics of the group are contained in the set *interests* and the access type of the group is equal to the value of the parameter *accessibility*.

A software agent a_n is also associated with each node of N , able to perform the following two tasks:

- It can be delegated by its node n to form a group of nodes, based on a set of requirements RSET. To this aim, the agent of n sends a joining request to all the nodes belonging to the admissible region $S(RSET)$, determined by using the *HySoN* protocol (see Section 4). The joining request is a tuple $JR = \langle t, a \rangle$, where t is a set of topics associated with the group and a is the access type of the group. After this joining, it periodically verifies that at least the α percent of the group members' profiles satisfy at least the β percent of the requirements of RSET. If this verification fails for a given user u_m , the agent of user u_n deletes the user u_m from the group, eventually notifying u_m itself of the removal.
- It negatively responses to a request of joining $JR = \langle t, a \rangle$ with a group coming from another agent a_m if the parameter *availability* = *false*. Otherwise, if *availability* = *true*, it positively responses if both the topics t of the group are contained in the set *interest* and the access type a of the group is equal to the value of the parameter *accessibility*.

4 The HySoN Protocol

According to the schema reported in the Section above, any agent can be delegated by its user to form a new group basing on some properties of interest. Therefore the delegated agent will look for a set of agents representing nodes which satisfy the set of requirements *RSET*. We are considering a large and very dynamic social network, e.g. Facebook, and the data to be analysed by the software agents is distributed over the users machines, therefore to support such a finding process we adopt a *peer-to-peer* approach since it is known to be more efficient and scalable than a centralised solution [26,15].

The finding schema adopted in this work is derived from a fully decentralised resource finding approach for large scale distributed systems developed and studied by the authors in the recent years [34,35,37], and is based on the construction of an overlay network of software agents which is, in fact, the building base of the *HySoN* protocol. As we discuss in Section 4.2, the *HySoN* topology reflects the distribution of the selected properties over the users. This characteristic, as we show in Section 5, is the basis for an efficient finding process.

The *HySoN* protocol includes three phases which are depicted in Figure 1. Once user data have been analysed, the delegated agent starts with the dissemination (dashed arrow (1) in Figure 1) of a gossip message starting from its neighbours; the dissemination phase is detailed in Section 4.1.

The *HySoN* overlay construction (dashed arrow (2) in Figure 1) is then executed by the agents already reached by the gossip message, by exploiting the existing links of the social network, as in the dissemination phase. This phase is discussed in Section 4.2.

The algorithm used for the *HySoN* construction is also the basis for the adopted finding process, i.e the third phase (dashed arrow (3) in Figure 1), and is discussed in Section 4.3.

4.1 Disseminating HySoN Overlay Construction Request

In the proposed model it is assumed that whenever two users are friends in the social network, their agents are able to communicate and exchange information. Therefore, the overlay topology used in the dissemination phase reflects the friendship of the social network. The dissemination phase is started whenever an agent is delegated by its user to start the construction of the *HySoN* overlay network and is explained below:

- The user agent sends a “gossip” message to its own neighbours (i.e. the software agent connected to its own friends).
- The gossip message contains a list of properties (p^1, p^2, \dots, p^l) to be used by each agent to compute and expose its own coordinates into the *HySoN* hyperspace (see Section 4.2).
- Once the gossip message is received by the generic agent n , user data are analysed to obtain values for the properties (p^1, p^2, \dots, p^l) specified in the message; then it forwards the gossip message to a subset of its neighbours, as specified in the gossip algorithm described in Section 5.
- Finally, to start with the second phase (overlay construction), agent n connects itself to the sender and starts running the *HySoN* overlay construction algorithm as described in Section 4.2.

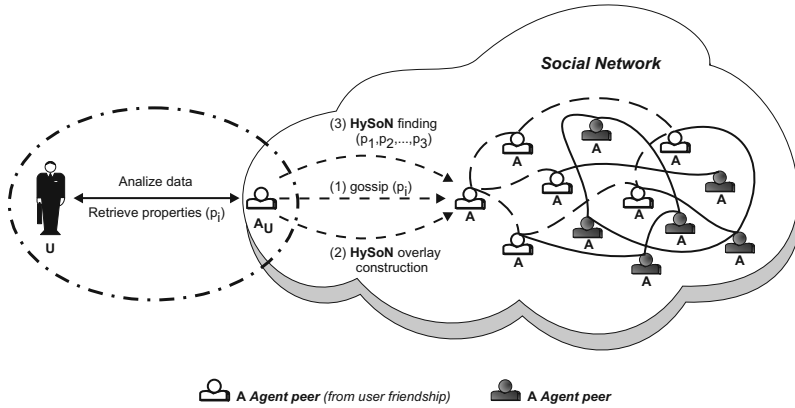


Fig. 1. The HySon Protocol

It should be noted that phase 1 (dissemination) and phase 2 (overlay construction) partially overlap, i.e. while the gossip message is still spreading into the social network, the agents already reached by the gossip are executing the overlay construction algorithm, which is described in the Section below.

4.2 HySoN Construction Algorithm

The HySoN overlay construction is the key for a fast and effective fully decentralised finding process, and it is performed by means of a decentralised algorithm which runs on the agents reached by the gossip message.

As specified in the description of the dissemination phase it is assumed that the generic agent n has been reached by the gossip message and has already retrieved the values correspondent to the requested properties. Property values are then mapped into “agent coordinates” in a multidimensional space (i.e. the hyperspace), and can be used to compute the Euclidean distance between agents. The overlay construction algorithm is listed below:

1. Agent n computes the set of agents at 2-hops, say N' ; for this aim, it asks to its neighbours (say N the set) the lists of their own neighbours.
2. The set N' is ordered by using the Euclidean distance of each agent from n .
3. Agent n selects from N' at least deg_{min} near agents, but at most deg_{trg} agents, obtaining the new set of neighbours, say N'' .
4. The agent connects to the peers belonging to N'' but not already in N ; moreover, the connection to the agents in N but not in N'' are removed.

Moreover, as detailed in [35], during the last step of the construction algorithm, the so-called *essentially critical* neighbours are preserved to make sure that the overlay network stays connected during the process.

The effect of the steps above is to create *clusters of agents* featuring a short intra-cluster distance, while keeping *long links* (i.e. the links preserved for the essentially critical neighbours) between clusters. As proved in [35] this resulting network features a structure quite similar to a *small-world* [48], i.e. a network which shows a high clustering degree and a very low average path length; these characteristics, as shown in Section 5, are very important to make resource finding effective.

Since the Euclidean distance is a similarity measure for the properties mapped on the coordinates, the clusters are characterised by agents exposing properties very closed to each other. Furthermore, by exploiting short links, a fast navigation inside the cluster is possible to e.g. refine the finding process, while, by using long links, the region (i.e. the cluster) close to the admissible region $S(RSET)$ (see next Section) can be quickly reached.

Some steps in the construction of an overlay network – in this case two properties/attributes have been mapped – are depicted in Figure 2.

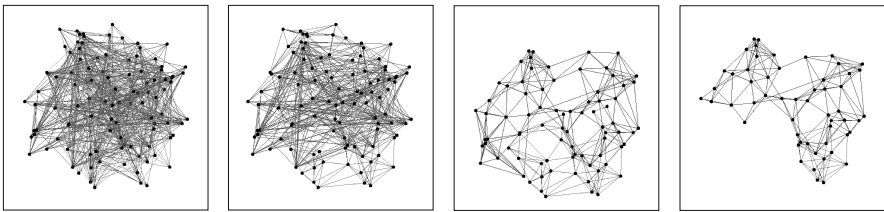


Fig. 2. The overlay network construction

4.3 Finding Suitable Nodes with HySoN

According to the said hyperspace abstraction, not only the single agent can be viewed as a point in the metric space, but also a *finding request* can be represented in the same way. Such a request carries a set of requirements ($RSET$) on the set of properties which are mapped as coordinates of the hyperspace.

In this way the request denotes a *point* representing a corner of a *region* or *semi-space* whose internal agents are those that expose suitable properties for the given finding request. Such a semi-space is called the *admissible region* $S(RSET)$.

The finding process is based on a fully decentralised finding algorithm, which is based on the following *check-and-forward* model:

1. an agent receiving the request checks if its properties satisfy the set of requirements in $RSET$; if this is the case, the agent belongs to the admissible region and the finding process continues with step 4;
2. if the set of requirements $RSET$ does not hold for the properties held by the agent, the request is submitted to one of its neighbours, which is selected on the basis of appropriate heuristics which allow the request to reach the admissible region as soon as possible [37];

3. the algorithm keeps track of all the peers visited (this set is carried together with the request); in this way there are no loops and duplicate messages;
4. when a peer belonging to the admissible region has been found, the other valid peers can be reached by suitably navigating through links [37] in order to build the needed set for the group formation.

As we simulated all the phases of the proposed approach (i.e. dissemination, overlay construction and finding process), we discuss the experimental results in the next Section.

5 Experimental Results

We evaluated the proposed approach in a social network of 10^6 users through the ComplexSim simulation platform [36,38]. For this aim, the topology of the social network was based on the well known scale free model proposed by A. Barabasi [5].

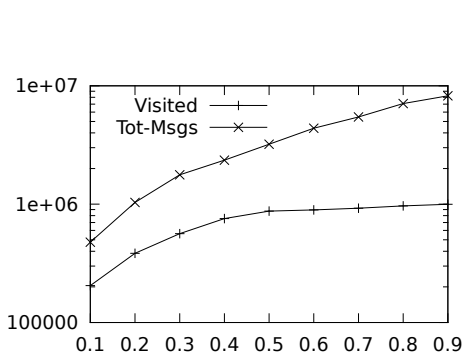
For the first phase (dissemination of the overlay network construction request), a probabilistic gossip protocol was employed. It is summarised in the listing 3b (function *check_gossip()*), while listing 3a reports a frame of the behaviour of the HySoN agent, which shows the receiving of the message, the call to the function *check_gossip()* and the searching function (the latter has been detailed in Section 4.3 but the pseudo-code is not shown here).

The schema of the gossip protocol is very simple, as the behaviour of the function *check_gossip()* is basically tuned by the threshold parameter v . When v is close to 1, the gossip message is likely to be propagated to the whole agent's neighbourhood; in this case the *hubs* of the network generates, in average, huge bags of messages, but it can involve in an excessive overhead. Conversely, in order to reach most of the nodes, the threshold v should not be too low. Furthermore, a TTL (Time-To-Live) and a message cache – the latter is maintained by each agent – are needed to stop the process in a few steps and to limit the number of duplicated messages.

<pre> //Agent (n) behaviour // ... msg = receive_msg(n); if (check_gossip(msg,n,v)==1) // Starts the hyson clustering //The sender is the first neighbour hyson(msg.sender, msg.properties); else if (is_finding (msg,n)==1) // distributed search //algorithm search(msg.request) </pre>	<pre> int check_gossip(msg,n,v){ if (! is_gossip (msg)) return 0; if (!(is_cached (msg,n) OR msg.ttl == 0)) put_msg_in_cache (msg,n); m = create_msg(msg.ttl -1) for all nn in neighbours(n) if random_uniform(0,1) < v send(m, nn) return 1; else return 0; } </pre>
---	---

Fig. 3. a) Agent behaviour

b) Gossip function



$deg_{min} = 6, deg_{trg} = 8$

T	Density	Degree			l	C
		min	max	avg		
0	2.0e-6	1	21	2.0	19.4	1e-6
1	9.7e-6	6	147	9.7	24.6	0.38
...						
10	1.0e-5	6	41	10.4	13.5	0.20

$deg_{min} = 18, deg_{trg} = 20$

T	Density	Degree			l	C
		min	max	avg		
0	2.0e-6	1	34	2.0	17.4	1e-6
1	2.9e-5	18	981	29.1	9.6	0.25
...						
10	3.3e-5	18	188	32.8	4.4	0.24

Fig. 4. a) HySoN request Dissemination b) Main characteristics of the HySoN topology (T=time-step, l=Geodesic Path, C=clustering coefficient)

The experimental results (in logarithmic scale) of the dissemination phase – in terms of number of nodes reached by the gossip, and generated messages, for different values of the threshold ν (see Listing 3b) and $TTL = 3$ – are reported in Figure 4a. We note that when the threshold span in the range $0.4 \div 0.5$, about the 80% of nodes can be reached with no more than (about) $3n$ messages (where n is the total number of nodes).

Table 4b gives the main characteristics of the HySoN overlay network – i.e. the result of the second phase – in terms of average minimum path (see index l) and clustering coefficient (see parameter C) for two couples of degree thresholds (parameter deg_{min} and deg_{trg} , discussed in Section 4.2). These data make evidence that the resulting overlay network exhibits the small world properties (i.e. low geodesic path, l , and high clustering coefficient, C). We also observed that the HySoN clustering algorithm leads to a stable network after a few steps (about 10).

In the end we carried a set of simulation of the finding algorithm in the resulting network (third phase), measuring the total number of time-steps to reach the admissible region $S(RSET)$. The results are shown in Figure 5. In this case the coordinates of the nodes were appropriately tuned for the different experiments; in this way the ratio of suitable nodes (x-axis in Figure 5), i.e. nodes which can satisfy the constraints of the finding requests, assumed the desired range.

As depicted in Figure 5, the finding algorithm performs well in terms of time-steps (see first quartile, median and third quartile) even when the ratio of suitable nodes is rather low (i.e. $0.1 - 0.2$). This result is due to the fact that the agents holding similar properties are connected with high probability, while the average geodesic path is maintained small enough.

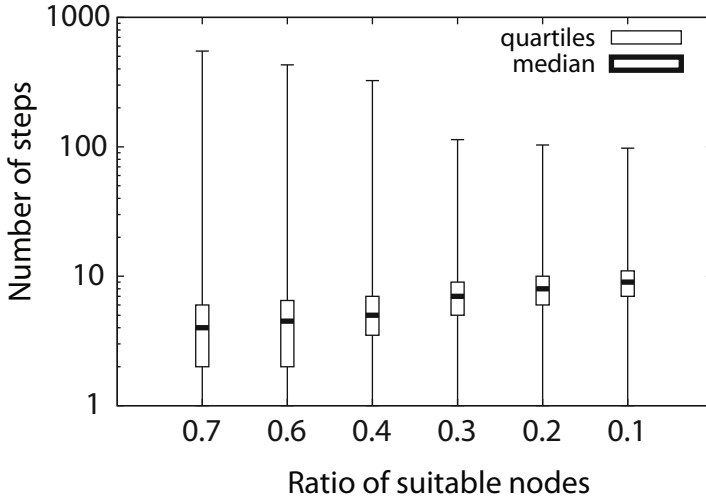


Fig. 5. Number of steps to reach the admissible region basing on the requirements RSET

Basing on the last results we can remark that the cost due to the construction of the *HySoN* network shown in Figure 4a is compensated not only by the privacy improvements for the users (we discuss this concern in Section 2 and 3), but also by the performance of the finding process. Indeed, once the network has been constructed, multiple requests can be sent by different agents according to additional requirements provided by their own users.

6 Conclusions and Future Works

In on-line social networks, an important issue is that of preserving the users' privacy. However, the possibility for the users of finding new profitable contacts depends on information which the user is willing to share on the network. In this paper, we have presented an agent-based mechanism to support the users of an on-line social network in the formation of groups composed of users having similar interests. This approach allows the users to locally maintain sensitive information, on the one hand, and to offer an efficient mechanism for each user to reach promising interlocutors, on the other hand. The agents analyse the local user's information in order to extract relevant properties which are then exploited to find the appropriate interlocutors. The set of the interlocutors is organised into a custom overlay network called *HySoN* (Hyperspace Social Network), built by exchanging mutual messages between agents using a gossip protocol.

The so-built network provides the users with the possibility to send their requests over the network in order to find a set of agents having some required properties, in order to form a group of users.

We have shown by experiments performed on a large network of simulated users that our approach is capable of reaching approximately all the agents in a small number of

steps and with a limited number of exchanged messages, and that the search of suitable nodes is very efficient due to the topology of the network, which exhibits the small-world properties. The main advantage of our approach is that the properties involved in the user's aggregation are inferred by local data not published in the social network, thus preserving the users' privacy with respect to their sensitive data.

As for our ongoing research, we are planning to introduce a trust mechanism in our approach, as for instance in [33], that will use trust measures in order to limit the effects of misbehaving nodes in the formation of groups.

Acknowledgements. This work is a part of the research project **PRISMA**, code **PON04a2_A/F**, funded by the Italian Ministry of University within the **PON 2007-2013** framework program.

References

1. Anagnostopoulos, A., Becchetti, L., Castillo, C., Gionis, A., Leonardi, S.: Power in unity: forming teams in large-scale community systems. In: Proc. 19th ACM Int. Conf. on Information and Knowledge Management, pp. 599–608. ACM (2010)
2. Anagnostopoulos, A., Becchetti, L., Castillo, C., Gionis, A., Leonardi, S.: Online team formation in social networks. In: Proc. 21st Int. Conf. on WWW, pp. 839–848. ACM (2012)
3. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., et al.: Above the clouds: A Berkeley view of cloud computing. EECS Dept., University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28 (2009)
4. Backstrom, L., Huttenlocher, D., Kleinberg, J., Lan, X.: Group formation in large social networks: membership, growth, and evolution. In: Proc. 12th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 44–54. ACM (2006)
5. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. *Science* 286(5439), 509–512 (1999)
6. Berkhin, P.: A survey of clustering data mining techniques. In: *Grouping Multidimensional Data*, pp. 25–71. Springer (2006)
7. Brandes, U., Gaertler, M., Wagner, D.: Engineering graph clustering: Models and experimental evaluation. *ACM J. of Experimental Algorithmics* 12(1.1), 1–26 (2007)
8. Buccafurri, F., Palopoli, L., Rosaci, D., Sarné, G.M.L.: Modeling cooperation in multi-agent communities. *Cognitive Systems Research* 5(3), 171–190 (2004)
9. Bulatov, A.A.: Complexity of conservative constraint satisfaction problems. *ACM Transactions on Computational Logic (TOCL)* 12(4), 24 (2011)
10. Burt, R.S.: *Structural holes: The social structure of competition*. Harvard Univ. Press (2009)
11. Carrington, P.J., Scott, J., Wasserman, S.: *Models and methods in social network analysis*. Cambridge Univ. Press (2005)
12. Castells, M.: *The Internet galaxy: Reflections on the Internet, business, and society*. Taylor & Francis (2003)
13. Chhabra, M., Das, S., Szymanski, B.: Team formation in social networks. In: *Computer and Information Sciences III*, pp. 291–299. Springer (2013)
14. Costa, P., Napper, J., Pierre, G., van Steen, M.: Autonomous resource selection for decentralized utility computing. In: 29th IEEE Int. Conf. on Distributed Computing Systems, ICDCS 2009, pp. 561–570. IEEE (2009)
15. Talia, D., Trunfio, P.: Toward a synergy between p2p and grids. *IEEE Internet Computing* 7(4), 94–96 (2003)

16. Ferber, J.: *Multi-agent systems: an introduction to distributed artificial intelligence*, vol. 1. Addison-Wesley Reading (1999)
17. Furht, B., Escalante, A.: *Handbook of cloud computing*. Springer (2010)
18. Gaertler, M.: Clustering. In: Brandes, U., Erlebach, T. (eds.) *Network Analysis*. LNCS, vol. 3418, pp. 178–215. Springer, Heidelberg (2005)
19. Gajewar, A., Sarma, A.D.: Multi-skill collaborative teams based on densest subgraphs. In: *SDM 2012* (2011)
20. Gaston, M.E., desJardins, M.: Agent-organized networks for dynamic team formation. In: *Proc. 4th int. Conf. on Autonomous Agents and Multiagent Sys.*, pp. 230–237. ACM (2005)
21. Giunta, R., Messina, F., Pappalardo, G., Tramontana, E.: Providing qos strategies and cloud-integration to web servers by means of aspects. *Concurrency and Computation: Practice and Experience* (2013), doi:10.1002/cpe.3031
22. Holme, P., Newman, M.E.J.: Nonequilibrium phase transition in the coevolution of networks and opinions. *Physical Review E* 74(5), 056108 (2006)
23. Hopcroft, J., Khan, O., Kulis, B., Selman, B.: Tracking evolving communities in large linked networks. *Proc. of the National Academy of Sciences of the USA* 101(sup. 1), 5249–5253 (2004)
24. Howard, B.: Analyzing online social networks. *Comm. of the ACM* 51(11), 14–16 (2008)
25. Huebscher, M.C., McCann, J.A.: A survey of autonomic computing: degrees, models, and applications. *ACM Computing Surveys* 40(3), 7 (2008)
26. Iamnitchi, A., Foster, I.: A peer-to-peer approach to resource location in grid environments. In: *Grid Resource Management*, Kluwer Pub. (2003)
27. Jackson, M.O.: A survey of network formation models: Stability and efficiency. In: *Group Formation in Economics: Networks, Clubs and Coalitions*, pp. 11–57 (2005)
28. Kannan, R., Vempala, S., Vetta, A.: On clusterings: Good, bad and spectral. *J. of the ACM (JACM)* 51(3), 497–515 (2004)
29. Klusch, M.: Information agent technology for the internet: a survey. *Data & Knowledge Engineering* 36(3), 337–372 (2001)
30. Lappas, T., Liu, K., Terzi, E.: Finding a team of experts in social networks. In: *Proc. 15th SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pp. 467–476. ACM (2009)
31. Li, C.T., Shan, M.K.: Team formation for generalized tasks in expertise social networks. In: *2010 IEEE 2nd Int. Conf. on Social Computing (SocialCom)*, pp. 9–16. IEEE (2010)
32. Marx, K., Bottomore, T.B., Rubel, M., Fromm, E.: *Selected writings in sociology & social philosophy*. McGraw-Hill, New York (1964)
33. Messina, F., Pappalardo, G., Rosaci, D., Santoro, C., Sarné, G.M.L.: A trust-based approach for a competitive cloud/Grid computing scenario. In: Fortino, G., Badica, C., Malgeri, M., Unland, R. (eds.) *Intelligent Distributed Computing VI*. SCI, vol. 446, pp. 129–138. Springer, Heidelberg (2012)
34. Messina, F., Pappalardo, G., Santoro, C.: Hygra: A decentralized protocol for resource discovery and job allocation in large computational grids. In: *2010 IEEE Symposium on Computers and Communications (ISCC)*, pp. 817–823. IEEE (2010)
35. Messina, F., Pappalardo, G., Santoro, C.: Exploiting the Small-World Effect for Resource Finding in P2P Grids/Clouds. In: *Proc. 20th IEEE Int. Work. on Enabling Technologies: Infrastructures for Collaborative Enterprises*, pp. 122–127 (2011)
36. Messina, F., Pappalardo, G., Santoro, C.: Complexsim: An smp-aware complex network simulation framework. In: *2012 6th Int. Conf. on Complex, Intelligent and Software Intensive Systems (CISIS)*, pp. 861–866. IEEE (2012)
37. Messina, F., Pappalardo, G., Santoro, C.: Decentralised resource finding in cloud/grid computing environments: A performance evaluation. In: *IEEE 21st Int. Work. on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pp. 143–148. IEEE (2012)

38. Messina, F., Pappalardo, G., Santoro, C.: Complexsim: a flexible simulation platform for complex systems. *International J. of Simulation and Process Modelling* (2013)
39. Perer, A., Shneiderman, B.: Balancing systematic and flexible exploration of social networks. *IEEE Transactions on Visualization and Computer Graphics* 12(5), 693–700 (2006)
40. Postmes, T., Spears, R., Lea, M.: The formation of group norms in computer-mediated communication. *Human Communication Research* 26(3), 341–371 (2000)
41. Rosaci, D., Sarné, G.M.L.: Matching users with groups in social networks. In: Zavoral, F., Jung, J.J., Badica, C. (eds.) *Intelligent Distributed Computing VII. SCI*, vol. 511, pp. 45–54. Springer, Heidelberg (2013)
42. Rosaci, D., Sarné, G.M.L., Garruzzo, S.: Integrating trust measures in multiagent systems. *International Journal of Intelligent Systems* 27(1), 1–15 (2012)
43. Sarkar, P., Moore, A.W.: Dynamic social network analysis using latent space models. *ACM SIGKDD Explorations Newsletter* 7(2), 31–40 (2005)
44. Schaeffer, S.E.: Graph clustering. *Computer Science Review* 1(1), 27–64 (2007)
45. Sofia Pereira, C., Soares, A.L.: Improving the quality of collaboration requirements for information management through social networks analysis. *Int. J. of Information Management* 27(2), 86–103 (2007)
46. Steinhäuser, K., Chawla, N.V.: Identifying and evaluating community structure in complex networks. *Pattern Recognition Letters* 31(5), 413–421 (2010)
47. Wasserman, S., Faust, K.: *Social network analysis: Methods and applications*, vol. 8. Cambridge univ. press (1994)
48. Watts, D., Strogatz, S.J.: Collective Dynamics of ‘Small-World’ Networks. *Nature* 393(6684), 440–442 (1998)
49. Yang, W.S., Dia, J.B., Cheng, H.C., Lin, H.T.: Mining social networks for targeted advertising. In: *Proc. 39th Annual Hawaii Int. Conf. on System Sciences, HICSS 2006*, vol. 6, p. 137a. IEEE (2006)

Intelligent Jason Agents in Virtual Soccer Simulations

Dejan Mitrović¹, Mirjana Ivanović¹, and Hans-Dieter Burkhard²

¹ Department of Mathematics and Informatics, Faculty of Sciences,
University of Novi Sad, Serbia
{dejan,mira}@dmi.uns.ac.rs

² Humboldt University, Institute of Informatics,
Rudower Chaussee 25, D-12489 Berlin, Germany
hdb@informatik.hu-berlin.de

Abstract. The game of virtual soccer requires its players to act as a coherent team, while operating in a highly dynamic environment, with incomplete and unreliable information. As such, the game is often used to test, demonstrate and validate the concepts of multi-agent theory in practice. This paper describes the process of integrating agent-oriented programming language *AgentSpeak* and its interpreter *Jason* into *SimSpark*, the official *RoboCup* soccer simulator. The end-goal is to design a framework that will enable its users to think about and program soccer playing agents at a more abstract level, in terms of beliefs, goals and plans, allowing for an easier implementation of advanced multi-agent concepts.

1 Introduction

Video games are often found at the forefront of artificial intelligence (*AI*) research, steering the development of various *AI* concepts and methodologies, including neural networks, genetic algorithms, case-based reasoning and intelligent agents [3, 14, 15]. Besides the entertaining factor, games can be efficiently used to model environments and scenarios that are hard/interesting from the *AI* research point of view. The game of football or *soccer*, for example, can be used to model *dynamic* environments with *unreliable* and *incomplete* information, in which a *team* effort is required to win [27].

RoboCup is an annual internationally-recognized competition of (primarily) soccer playing robots [31]. It comprises several leagues, including *simulation*, in which the main focus is on *AI* algorithms and techniques, and *humanoid*, which additionally deals with physical aspects of robotics. The main objective of *RoboCup* is "to promote robotics and *AI* research, by offering publicly appealing, but formidable challenge" [31]. In accordance to this objective, for example, each competing team is required to release its source code at the end of the competition, while the game is continuously made more difficult.

Software agents represent one of the most consistent approaches to distributed *AI*. In the field of soccer playing robots, the term *agent* is often used to describe the software system that performs deliberation and controls the robot's physical

parts. As discussed later, advanced concepts of the agent technology play an important role in modern soccer simulations.

In order to simplify the development of complex deliberate agents, a number of *agent-oriented* programming languages has been developed. Among the most popular languages is *AgentSpeak*, accompanied by the *Jason* interpreter [8,9,22]. This paper presents the initial work on programming soccer playing agents using *AgentSpeak*. To achieve this goal, a new development framework is proposed. It includes an extended version of the *Jason* interpreter, capable of deploying and running agents in *SimSpark* – the official *RoboCup* simulator [34]. The use of *AgentSpeak* and *Jason* as the basis of this framework will allow for a more abstract application of advanced concepts of the agent technology. *AgentSpeak* allows developers to think about and program agents in terms of beliefs, goals, and plans. Therefore, the proposed framework might represent an excellent tool for developing efficient soccer playing agents.

During this development, our main concern was the run-time efficiency of *Jason* agents. The *SimSpark* simulator is designed to operate in 20 millisecond cycles. Within this time-frame the agent needs to complete its entire *reasoning cycle*: sense the environment, decide on the next step(s), and perform the desired action(s). If the agent fails to respond in a timely manner, it might overlook an important event, end-up with an invalid view of the actual game state, or try to perform an action that is no longer valid. Performance evaluation results will show that the proposed framework is capable of running agents within the set time constraints. Nonetheless, some suggestions for future code optimizations are given in the concluding section.

The rest of this paper is organized as follows. Section 2 describes the functionalities of *SimSpark* and *Jason*, and highlights their similarities. Section 3 provides the details on *Jason-SimSpark* integration, and presents a case-study along with evaluation results. Existing work related to the usage of agents in virtual soccer simulations is analyzed in Section 4. Finally, general conclusions and future research directions are given in Section 5.

2 Framework Overview

In order to integrate *Jason* agents into *SimSpark*, it is necessary to analyze and understand the functioning of each framework. This section provides details on inner workings of *SimSpark* and *Jason*, focusing on the concepts that are important for their efficient integration.

2.1 SimSpark

SimSpark represents an advanced virtual environment for physical simulations of multi-agent systems [34]. It incorporates an accurate simulator for a range of physical properties, such as gravity, inertia, and collision detection, and provides a 3D visualization tool. *SimSpark* is the official simulator for the *RoboCup 3D Soccer Simulation League* [6], but it can also be adapted for various other multi-agent scenarios, such as modeling of disaster management.

SimSpark operates as a multi-agent environment in which an agent interacts with its surroundings as well as other agents. This interaction is achieved through the use of *effectors* and *perceptors* [7]. In essence, an effector is a command that the agent sends to the simulator in order to perform the desired action. Examples include bending the knee (or any joint), saying something to other agents, or setting the player’s position before the game starts. The set of available effectors is not fixed, and can easily be extended to support application-specific requirements.

To sense their environment, agents in *SimSpark* rely on perceptors. The simulator generates a set of perceptors at regular time intervals, and sends them to running agents. A perceptor can carry some general information, such as the force that acts on the agent’s body, or application-specific details, including the ball position relative to the agent’s camera, or current status of the game. That is, as with effectors, new perceptors can be included to satisfy requirements of the simulated scenario.

At runtime, the simulation engine of *SimSpark* is executed in a continuous loop. Each loop cycle consists of the following events [7]:

1. Start the cycle;
2. Send perceptors to each running agent;
3. Apply effectors received during the previous cycle;
4. Apply laws of physics and any rules of the game being played;
5. Finalize the cycle.

The described process is presented graphically in Fig. 1. It is important to note that the effectors an agent sends in cycle n are actually applied in cycle $n + 1$, and can be observed by the agent no earlier than cycle $n + 2$.

The duration of each cycle is set to 20 milliseconds, and it is crucial for the agent to complete its reasoning cycle within the given time-frame. By default, *SimSpark* will not wait for any agent to actually send its effectors. For development and debugging purposes, however, the simulator can be configured not

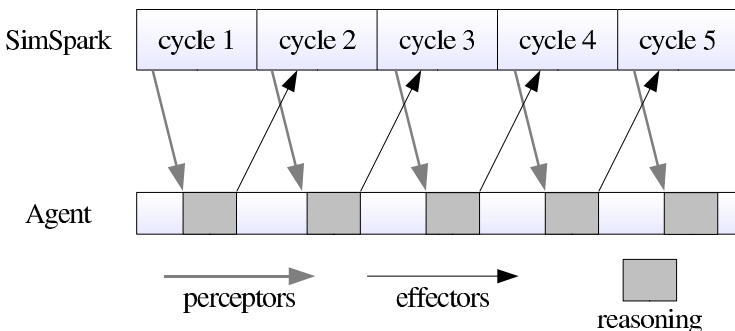


Fig. 1. Synchronization between the execution loop of *SimSpark* and an agent (adapted from [7])

advance to the next cycle until all running agents have sent their respective effectors.

Agents can be deployed to *SimSpark* in two ways: as internal plug-ins, or as external, stand-alone software components. In the latter case, the interaction between agents and the *SimSpark* server is performed through *TCP* and *UDP* network protocols. Although the plug-in based approach offers better performance, the development of external agents is more flexible, as any modern programming language can be used. The framework presented in this paper supports external *SimSpark* agents, which is also the default approach in *RoboCup* competitions [6].

2.2 AgentSpeak and Jason

AgentSpeak is a logic-based agent-oriented programming language, initially proposed in [28]. The language is designed for developing primarily *BDI* agents [10, 29, 35]. *AgentSpeak* uses modal logics to provide formalisms for the *BDI* architecture, and was originally defined as an abstract language. However, due to its elegance, the language has relatively recently been extended with several practical concepts, and incorporated into a modern interpreter named *Jason* [8, 9, 22].

Jason agents are defined in terms of *beliefs*, *goals*, and *plans*. Beliefs include statements the agent considers to be true, whereas goals describe the future state of affairs it wants to achieve. In this sense, beliefs and goals express the agent's *mental attitude*. Plans, on the other hand, define the means for fulfilling desired goals.

In practice, agent's beliefs are expressed as first-order logic formulae, with the syntax strongly resembling *Prolog* facts and rules. *Jason* extends this syntax with the concept of *annotations*. Annotations do not improve the expressiveness of the language, but provide an elegant way of associating additional details with a belief. For example, the fact *ball(10, 10)[source(percept)]* describes that the agent believes the ball is at position (10, 10), and has obtained this belief by sensing the environment. Additionally, *Jason* provides the strong negation operator to define statements the agent explicitly believes to be false.

Jason supports two types of goals: *achievement* and *test*. Achievement goals drive the agent's behavior, by triggering execution of plans. For example, the goal *!kick(ball)* might invoke a plan that would bend the agent's knee and perform the kick. Test goals are generally used to query the belief base. For example, goal *?see(ball)* can be used to test if the agent actually sees the ball.

A plan is defined in form of *triggeringEvent* : *context* \leftarrow *body*. The triggering event describes changes in the agent's mental attitude that should invoke the plan execution. Before the execution actually starts, however, *Jason* interpreter validates conditional statements specified in the plan context. This validation process is especially useful in dynamic environments, and is used to increase the chances of a successful execution of the plan. Once the triggering event occurs and the context is validated, the agent starts executing the sequence of formulae defined in the plan body.

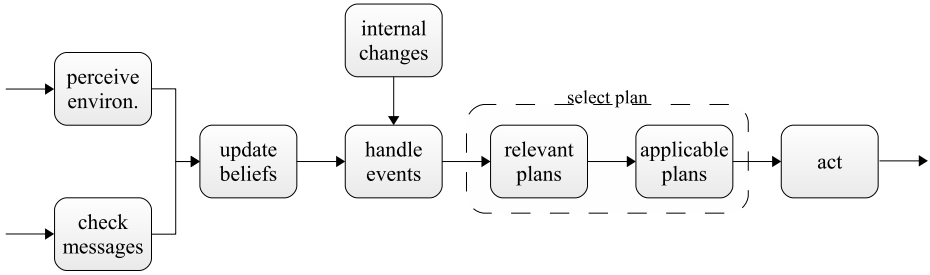


Fig. 2. Simplified outline of a single reasoning cycle in *Jason* (adapted from [9])

At runtime, the agent code written using the constructs described above is executed by the *Jason interpreter*. The interpreter operates in a loop, in terms of *reasoning cycles* [9]. A single reasoning cycle is shown in Fig. 2. The figure includes only a sub-set of the whole reasoning cycle, in order to better depict the functionalities that are of the special importance for the work presented in this paper; for the full analysis of the *Jason* reasoning cycle, see e.g. [9].

Jason interpreter executes the following algorithm in each reasoning cycle [9]:

1. Perceive the environment;
2. Handle a single message received from other agent;
3. Update the belief base, by processing percepts and the received message;
4. Generate events in response to previous steps, or some internal changes (e.g. changes in the agent's goal base);
5. Select *relevant* plans for a single event. Analyse the context of each relevant plan and produce the set of *applicable* plans;
6. Select a single applicable plan and execute its body.

The presented discussion on *SimSpark* and *Jason* execution/reasoning cycles reveals some important similarities between the two frameworks. More details about these similarities, integration of *Jason* agents into *SimSpark*, and the performance evaluation are given in the next section.

3 Jason Agents in SimSpark

From the analysis given in the previous section it can be concluded that both *Jason* and *SimSpark* support agents that operate in *sense-think-act* cycles: the agent first senses its environment, then reasons about the next step(s), and finally performs some action(s). Additionally, both *Jason* and *SimSpark* are designed to be extensible, and allow for modification of existing, and addition of new functionalities. As a result, the process of integrating *Jason* agents into *SimSpark* does not pose significant technical problems.

As noted earlier, the default approach of deploying agents in *SimSpark* is to develop them as external plug-ins that interact with the simulator over a

network connection. The network communication protocol relies on *symbolic*, or *S-expressions* to encode perceptors and effectors [7]. *S-expression* is a string-based data structure well-suited for representing binary trees. It is a human-readable and compact data structure and, most importantly, can be parsed very efficiently.

In order to enable the deployment of *Jason* agents in *SimSpark*, several extensions to the *Jason* interpreter were required. The extensions include *S-expression* parser and generator, as well as a customized agent architecture, a set of internal actions, and a customized execution control [9]. Our implementation of parser and generator components is based on the implementation used in *magmaOf-fenburgh agent framework* [24] and released under *GNU General Public License*.

Implementation of the custom execution control was required in order to support *key-framed motions* of simulated robots. A key-framed motion consists of several important positions (i.e. *key frames*) which are interpolated over time to perform a motion. A motion usually needs to be performed during several reasoning cycles. However, a reasoning cycle in *Jason* does not have to end with an action; unless the agent explicitly sends an action at the end of the reasoning cycle, no effectors will be sent to *SimSpark*, and the running motion will fail. Therefore, the custom execution control performs a manual *flush* of any pending, motion-related effectors to *SimSpark*.

As noted earlier, a single *sense-think-act* cycle of an agent deployed in *SimSpark* needs to complete within the 20 millisecond time-frame. Therefore, a case-study has been developed in order to evaluate the run-time efficiency of *Jason* in the developed framework.

3.1 Run-Time Performance Evaluation

In order to assess the run-time performance of *Jason* in *SimSpark*, a simple soccer playing agent was implemented. The agent is designed as a state machine depicted in Fig. 3. In total, 6 states can be recognized:

- *Idle*: Waiting for the kickoff;
- *Searching ball*: The agent cannot see the ball. It keeps looking around and perceiving the environment until it does;
- *Walking to ball*: The agent sees the ball and walks towards it until it is close enough to perform a kick;
- *Kicking*: The agent is standing next to the ball and performing the kick motion. The ball can be kicked in any direction;
- *Done*: The ball is inside the goal;
- *Lying down*: The agent has fallen down.

The agent may fall at any time, especially if the implementation is too slow. In this situation, the agent's body and limbs are in a position different than what the agent believes, so a motion it tries to perform turns out to be invalid. Acceleration percepts are used to determine if the agent has fallen down and, if so, to initiate the stand up motion.

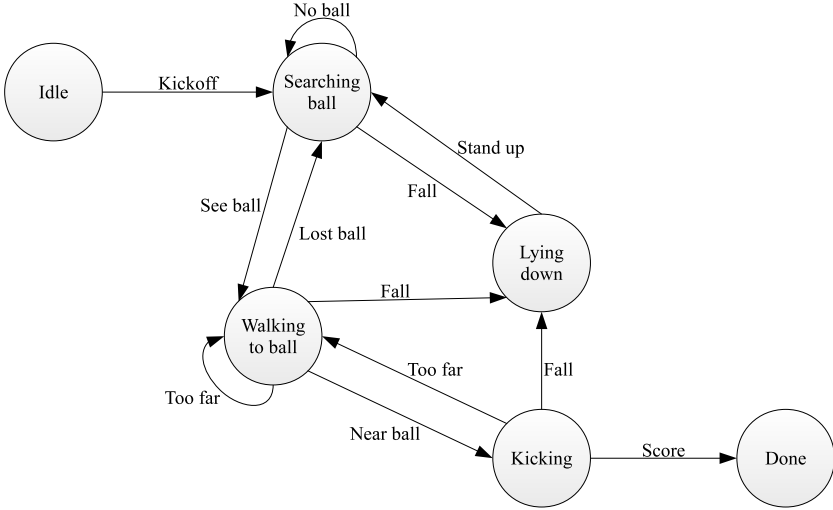


Fig. 3. State machine of the simple soccer playing agent

Listing 1.1 shows partial implementation of the agent. A number of actions were implemented to support key-framed motions, including *standUp*, *rollOver*, and *turnLeft*. Lines 2–5 define a set of rules to help determine if the agent is down, and if it’s close enough to the ball to perform the kick. To determine if it has fallen down, the agent examines forces that act on its body along the X , Y and Z -axis. This is achieved by analyzing the *acc* (shorter for *acceleration*) belief; if it’s standing straight, the agent will believe that $acc(0,0,9.81)$.

Once the kickoff is signaled, the agent starts searching for the ball. Lines 10–20 show several plans for accomplishing this. First of all, if the agent is down, it needs to stand back up. Otherwise, the agent turns around a bit and uses the test goal *?seeBall* to determine if it can now see the ball. If the test goal succeeds, the agent advances to the next state - walking towards the ball. If the test goal fails, so does the current plan, and the *Jason* interpreter generates goal deletion event $-!findBall$. In response, the agent tries to achieve goal *findBall* once again.

Listing 1.1. Partial implementation of the simple soccer playing agent in *AgentSpeak*

```

1 /* Initial beliefs and rules */
2 isDown :- acc(_, _, Z) & Z < 7.
3 isDownOnBack :- acc(_, Y, Z) & Z < 7 & Y > 0.
4 isDownOnFront :- acc(_, Y, Z) & Z < 7 & Y <= 0.
5 canKick(D) :- D < 0.1.
6
7 +kickOff : true <- +search. /* Initial goal */
8
9 /* Lying down */
10 +down : isDownOnBack <- standUp; -down; +search.
11 +down : isDownOnFront <- rollOver; standUp; -down; +search.
12 /* Searching for ball */
13 +search : true <- !findBall.
14 /* Walking towards the ball */
15 +walk : true <- !goToBall.
16

```

```

17 /* Plans for the achievement goal "findBall" */
18 +!findBall : isDown <- -search; +down.
19 +!findBall : not isDown <- turnLeft; ?seeBall(-, -, -); -search;
    +walk.
20 -!findBall : true <- !!findBall.

```

The run-time performance evaluation of the simple soccer playing agent was performed using the following hardware and software configuration:

- A quad-core *CPU* at 1.6GHz with 4GB of *RAM* running 64-bit version of *Microsoft Windows 7 Professional*;
- *Jason* version 1.3.9
- *SimSpark* version 0.6.4
- *Java 7*, 64-bit edition

The lower-end computer was selected because it would better demonstrate any loss of performance. During the experiments, the *SimSpark* monitor was disabled (i.e. only the simulator engine was running). Duration of the agent's reasoning cycles was measured over time, following the initial kickoff and including the next 5000 reasoning cycles. The kickoff is signaled 300 cycles after the application starts. Experiments involving 1, 2 and 6 soccer playing agents were conducted, with 6 being the default team size in *RoboCup* simulation league.

Fig. 4 shows the results of these experiments. Obviously, when hosting 1 or 2 agents, the framework performs very well, managing to execute agents' reasoning cycles within the given time-frame. When running 6 agents, the agents' reasoning cycles take longer to complete in the beginning, causing them to fall frequently. Eventually, however, the system stabilizes and the game continues as expected.

The presented results are very encouraging. They show that the run-time performance does not present an obstacle for using higher-level, agent-oriented programming languages. Backed-up by extensive existing research on using agents in soccer and other competitive games presented in the next section, these results provide a strong incentive for future improvements of the proposed framework.

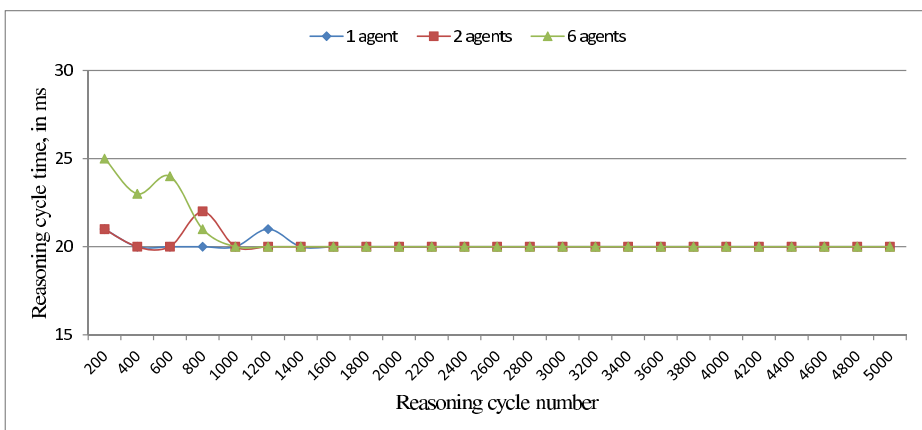


Fig. 4. Performance evaluation of the proposed framework

4 Related Work

Although many techniques of *AI* play an essential role in modern video games, adoption of the agent technology concepts has been minimal in the past. The use of *BDI* architectures in soccer simulations, for example, has mostly been limited to most basic skills [5, 11]. One of the main reasons for this situation is that video game developers tend to rely on *AI* techniques that provide reliable and predictable behavior [3, 15]. Adaptive *AI* techniques, on the other hand, do increase the sense of realism in games [3] and have recently started to receive a greater attention from game developers and researchers.

Adaptive techniques of the agent technology, such as agent cooperation, coordination, negotiation and *BDI* architectures have been especially useful in dynamic environments such as the game of soccer. A soccer playing agent usually has incomplete and unreliable information about the world, and needs to act together with other agents (i.e. its teammates) in order to win. With these properties, soccer is frequently used by researchers to test and demonstrate the concepts of agent theory in practice, often combining them with other general concepts of *AI* [4, 18, 23, 26, 27, 30, 32]. Additionally, it has been shown in [2] that agent-based soccer simulations can be efficiently used for educational purposes, i.e. to motivate undergraduate students and manage the complexity of robotics.

A review of the existing research focused on using agent coordination, cooperation and negotiation in virtual soccer simulations has been presented in [33]. The overall conclusion is that, although the agent theory fits nicely in the dynamic soccer environment, run-time efficiency remains an issue in most cases. On the other hand, our performance evaluation, although dealing with simple agents for now, shows that *Jason* represents an efficient run-time environment for intelligent agents playing soccer.

In general, *Java* represents the most widely used platforms for developing agents [10]. However, a special category of *agent-oriented* programming languages includes languages designed to simplify the development of complex, *reasoning* agents. *AgentSpeak* is an example of a *logic-based* agent-oriented language, best-suited for *BDI* agent architectures. Similar modern agent-oriented languages do exist, most popular of which are *GOAL* [16, 17] and *2APL* [1, 13]. There are several important reasons, however, why *AgentSpeak* was selected for the work presented in this paper. *AgentSpeak* code is executed by the efficient and extendible *Jason* interpreter, which, as presented, simplifies its integration with *SimSpark*. Secondly, both *AgentSpeak* and *Jason* are documented very well, and accompanied by a sufficient set of development and debugging tools.

One of the most decisive factors was, however, the relatively recent development of the *JaCaMo* platform [21]. *JaCaMo* integrates *AgentSpeak* and *Jason* with *CARTAGO* framework for artifacts modeling [12], and *Moise* framework for virtual organizations [25]. The usage of *Moise* for organizational structuring of soccer teams has already been proposed in [20]. Their work demonstrates how *Moise* provides an elegant solution for declaring multi-agent organizational structure at both the agent and the system level. Advanced features, such as dynamic reorganization in response to environmental changes, are also possible.

Jason and *Moise* have also been successfully used in a *cow-herding* programming contest [19]. The contest models a scenario in which teams of agents compete against each other for resources. The developed framework utilized a number of general *AI* concepts, and extended the *Jason* interpreter through custom agent architecture and action implementations, similarly to the work presented in this paper. Additionally, the reasoning time-frame of the underlying cow-herding simulator was limited to four seconds. The overall conclusion is that [19] the combined usage of *Jason* and *Moise* results in a very elegant framework: agent coordination and teamwork are managed by *Moise*, while the autonomy, pro-active behavior, and communication are handled by *AgentSpeak* and *Jason*. The framework is also run-time efficient, and was more than capable of satisfying the given time constraints.

This existing research on using agents, *Jason* and related frameworks in competitive, team-based games provides us with a strong incentive for future extension of the presented work, including an integration of the entire *JaCaMo* platform in the framework for intelligent soccer playing agents.

5 Conclusions and Future Work

Virtual soccer simulation represents an excellent environment for validating and demonstrating theoretical concepts of the agent technology in practice. This is because the game of soccer models a dynamic environment, with often incomplete and unreliable information, in which a team effort is required to win.

The work presented in this paper aims to design a framework suitable for developing intelligent, reasoning agents that operate in the official *RoboCup* simulator engine *SimSpark*. The main goal is to employ agent-oriented programming language *AgentSpeak* and its interpreter *Jason* in order to allow a more abstract usage of agent-based concepts. That is, the proposed framework enables elegant application of advanced concepts of the agent technology, and allows developers to think in terms of beliefs, goals, and plans, to employ the *BDI* agent architecture, as well agent communication and cooperation.

As shown, at run-time the soccer playing *Jason* agents perform very well, and are capable of completing their respective reasoning cycles within the *SimSpark*'s execution cycles. That is, for the described case-study, the *Jason* interpreter is efficient enough to satisfy strict time constraints imposed by *SimSpark*.

However, there is still plenty of room for optimizing the underlying code. For example, during the experiments it was observed that the *Jason* built-in function for generating belief literals from strings represents a performance bottleneck, especially because it is called often during a single reasoning cycle. The immediate future work will, therefore, be focused on identifying and removing existing performance bottlenecks.

In the long run, remaining parts of the *JaCaMo* platform – *Moise* and *CARTaGO* – will be integrated in the proposed framework. This will allow our soccer playing agents to exhibit more complex and truly intelligent behavior.

Acknowledgements. This work was partially supported by Ministry of Education, Science and Technological Development of the Republic of Serbia, through project no. OI174023: "Intelligent techniques and their integration into wide-spectrum decision support".

References

1. 2APL homepage, <http://apapl.sourceforge.net/> (retrieved on April 19, 2013)
2. Anderson, J., Baltes, J.: An agent-based approach to introductory robotics using robotic soccer. *International Journal of Robotics and Automation* 21(2), 141–152 (2006)
3. Bakkes, S.C., Spronck, P.H., van den Herik, H.J.: Opponent modelling for case-based adaptive game AI. *Entertainment Computing* 1(1), 27–37 (2009)
4. Benac Earle, C., Fredlund, L.-Å., Iglesias, J.A., Ledezma, A.: Verifying Robocup Teams. In: Peled, D.A., Wooldridge, M.J. (eds.) *MoChArt 2008*. LNCS, vol. 5348, pp. 34–48. Springer, Heidelberg (2009)
5. Berger, R.: Die doppelpass-architektur – verhaltenssteuerung autonomer agenten in dynamischen umgebungen. Diploma Thesis, Humboldt-Universität zu Berlin, Institut für Informatik (2006) (in German)
6. Boedecker, J., Asada, M.: SimSpark – concepts and applications in the RoboCup 3D soccer simulation league. In: *Workshop Proceedings of SIMPAR 2008, International Conference on Simulation, Modeling and Programming for Autonomous Robots*. pp. 174–181 (2008)
7. Boedecker, J., Dorer, K., Rollmann, M., Xu, Y., Xue, F., Buchta, M., Vatankhah, H.: SimSpark user’s manual (January 2010), <http://kent.dl.sourceforge.net/project/simspark/User>
8. Bordini, R.H., Hübner, J.F.: BDI agent programming in agentSpeak using *jason* (Tutorial paper). In: Toni, F., Torroni, P. (eds.) *CLIMA 2005*. LNCS (LNAI), vol. 3900, pp. 143–164. Springer, Heidelberg (2006)
9. Bordini, R.H., Hubner, J.F., Wooldridge, M.: *Programming multi-agent systems in AgentSpeak using Jason*. Wiley Series in Agent Technology. John Wiley & Sons Ltd. (2007)
10. Bădică, C., Budimac, Z., Burkhard, H.D., Ivanović, M.: Software agents: Languages, tools, platforms. *Computer Science and Information Systems* 8(2), 255–298 (2011)
11. Burkhard, H.-D., Bach, J., Berger, R., Brunswieck, B., Gollin, M.: Mental models for robot control. In: Beetz, M., Hertzberg, J., Ghallab, M., Pollack, M.E. (eds.) *Dagstuhl Seminar 2001*. LNCS (LNAI), vol. 2466, pp. 71–88. Springer, Heidelberg (2002)
12. CArtAgO homepage, <http://cartago.sourceforge.net/> (retrieved on April 19, 2013)
13. Dastani, M.: 2APL: a practical agent programming language. *International Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)* 16(3), 214–248 (2008)
14. Dignum, F., Bradshaw, J., Silverman, B., van Doesburg, W. (eds.): *Agents for Games and Simulations*. LNCS, vol. 5920. Springer, Heidelberg (2009)
15. Gabriel, I., Negru, V., Zaharie, D.: Neuroevolution based multi-agent system for micromanagement in real-time strategy games. In: *Proceedings of the Fifth Balkan Conference in Informatics, BCI 2012*, pp. 32–39. ACM, New York (2012)

16. GOAL homepage, <http://mmi.tudelft.nl/trac/goal> (retrieved on April 19, 2013)
17. Hindriks, K.V.: Programming rational agents in GOAL. In: El Fallah Seghrouchni, A., Dix, J., Dastani, M., Bordini, R.H. (eds.) *Multi-Agent Programming: Languages, Tools and Applications*, pp. 119–157. Springer, US (2009)
18. Hu, C., Mao, X., Zhou, H.: Programming dynamics of multi-agent systems. In: Kinny, D., Hsu, J.Y.-j., Governatori, G., Ghose, A.K. (eds.) *PRIMA 2011. LNCS*, vol. 7047, pp. 287–298. Springer, Heidelberg (2011)
19. Hübner, J.F., Bordini, R.H.: Using agent- and organisation-oriented programming to develop a team of agents for a competitive game. *Annals of Mathematics and Artificial Intelligence* 59(3-4), 351–372 (2010)
20. Hubner, J.F., Sichman, J.S., Boissier, O.: Developing organised multiagent systems using the moise+ model: programming issues at the system and agent levels. *International Journal of Agent-Oriented Software Engineering* 1(3/4), 370–395 (2007)
21. JaCaMo homepage, <http://jacamo.sourceforge.net/> (retrieved on April 19, 2013)
22. Jason homepage, <http://jason.sourceforge.net> (retrieved on April 19, 2013)
23. Leng, J., Fyfe, C., Jain, L.: Simulation and reinforcement learning with soccer agents. *Multiagent Grid Systems* 4(4), 415–436 (2008)
24. magmaOffenburg homepage, <http://robocup.fh-offenburg.de> (retrieved on April 19, 2013)
25. Moise homepage, <http://moise.sourceforge.net/> (retrieved on April 19, 2013)
26. Nisikata, T., Sawamura, H.: Deliberate soccer agents powered by resource-bounded argumentation. In: Shi, Z.-Z., Sadananda, R. (eds.) *PRIMA 2006. LNCS (LNAI)*, vol. 4088, pp. 656–663. Springer, Heidelberg (2006)
27. Pfeifer, J., Wainer, J.: Multi-agent plans: Analysis and implementation. In: Barros, L.N., Jr., R.M.C., Cozman, F.G., Costa, A.H.R. (eds.) *International Joint Conference IBERAMIA 2000 and SBIA 2000, Workshop Proceedings, Meeting on Multi-Agent Collaborative and Adversarial Perception, Planning, Execution, and Learning*, pp. 185–190 (2000)
28. Rao, A.S.: AgentSpeak(L): BDI agents speak out in a logical computable language. In: Perram, J., Van de Velde, W. (eds.) *MAAMAW 1996. LNCS*, vol. 1038, pp. 42–55. Springer, Heidelberg (1996)
29. Rao, A.S., Georgeff, M.P.: BDI agents: from theory to practice. In: Lesser, V., Gasser, L. (eds.) *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS 1995)*, pp. 312–319 (1995)
30. Raza, A., Sharif, U., Haider, S.: On learning coordination among soccer agents. In: *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 699–703 (2012)
31. RoboCup homepage, <http://www.robocup.org/> (retrieved on April 19, 2013)
32. Ruiz, M.A., Uresti, J.R.: Team agent behavior architecture in robot soccer. In: *Robotic Symposium, LARS 2008*, pp. 20–25. IEEE Latin American (2008)
33. Shukri, S.R.M., Shaukhi, M.K.M.: A study on multi-agent behavior in a soccer game domain. *World Academy of Science, Engineering and Technology* 14, 308–312 (2008)
34. SimSpark homepage, <http://simspark.sourceforge.net/wiki> (retrieved on April 19, 2013)
35. Wooldridge, M.J.: Reasoning about rational agents. *Intelligent Robotics and Autonomous Agents*. The MIT Press (2000)

Multi-agent Systems Applied in the Modelling and Simulation of the Protein Folding Problem Using Distributed Constraints

Ionel Muscalagiu¹, Horia Emil Popa², Manuela Panoiu¹, and Viorel Negru²

¹ The Faculty of Engineering of Hunedoara,

The "Politehnica" University of Timisoara, Revolutiei, 5, Romania

{ionel.muscalagiu,manuela.panoiu}@fih.upt.ro

² The Faculty of Mathematics and Informatics, The University of the West,

Timisoara, V.Parvan 4, Romania

{hpopa,vnegru}@info.uvt.ro

Abstract. The protein folding problem is one of the most challenging problems in current biochemistry and is an important problem in bioinformatics. All current mathematical models of the problem are affected by intrinsic computational limits. The previous research offers few approaches that make use of multi-agent systems to resolve this problem. In this paper we present an agent-based framework for protein structure prediction, composed by autonomous agents which collaborate in order to find a solution using Distributed Constraint Programming (DisCSP/DisCOP). Each amino acid of an input protein is viewed as an autonomous agent that communicates with others by transmitting messages.

In this article was analysed the NetLogo environment with the purpose of building a general model of implementation and simulation for the protein structure prediction. Starting from the proposed implementation model with lattice models based on distributed constraints, in this article we present a multi-agent systems which can be used for the implementation and simulation of the protein structure prediction, that can run on a single computer or on a cluster computing environment. The version of the tool presented herein allows studying and exploring complex problems belonging principally to structural biology, such as protein folding.

1 Introduction

Proteins are complex biological macro-molecules that are composed of a sequence of amino acids, which is encoded by a gene in a genome. There are 20 different amino acids specified in the genetic code. Proteins play key roles in many cellular functions. The functional properties of proteins depend upon their three-dimensional structures. Unlike the structure of other biological macromolecules, proteins have complex structures that are difficult to predict. The protein folding problem is one of the most challenging problems in current biochemistry and is a

very interesting problem in computational mathematics. The Protein Structure Prediction Problem is the problem of predicting the 2D/3D native conformation of a protein, when its sequence of amino acids is known [1],[2],[3],[8],[9],[18]. One of the most popular models of protein folding is the hydrophobic-hydrophilic (H-P) model [8],[9],[18]. In the H-P model, proteins are modelled as chains whose vertices are marked either H (hydrophobic) or P (hydrophilic); the resulting chain is embedded in some lattice. H nodes are considered to attract each other while P nodes are neutral. An optimal embedding is one that maximizes the number of H-H contacts.

Constraint programming is a programming approach used to describe and solve large classes of problems such as searching, combinatorial and planning problems. Distributed Constraint Satisfaction(DisCSP)/Distributed Constraint Optimization(DisCOP) is a framework for describing a problem in terms of constraints that are known and enforced by distinct participants (agents)[19],[12],[13],[17]. The constraints are described on some variables with predefined domains, and have to be assigned to the same values by the different agents. This type of distributed modelling appeared naturally for many problems for which the information was distributed to many agents.

Distributed constraints (DisCSP/DisCOP) are composed of agents, each owning its local constraint network. Variables in different agents are connected by constraints, agents must assign values to their variables so that all constraints between agents are satisfied (DisCSP) or, for DisCOP, a group of agents must choose values for a set of variables so that the cost of a set of constraints over the variables is either minimized or maximized. DisCOP search algorithms use search strategies to search through the solution space to find a cost-minimal solution.

There exist complete asynchronous searching techniques for solving the DisCSP in this constraints network, such as the ABT (Asynchronous Backtracking), AWCS (Asynchronous Weak Commitment) [19], ABTDO (Dynamic Ordering for Asynchronous Backtracking) [12] and DisDB (Distributed Dynamic Backtracking) [6]. Also, for DisCOP there are many algorithms between which we name ADOP (Asynchronous Distributed OPTimization) [13] or DPOP (Dynamic Programming Optimization Protocol) [17]. We find that many multi-agent problems can be reduced to a distributed constraints problem.

The previous research offers few approaches that make use of multi-agent systems to the purpose of proposing distributed solutions to the protein structure prediction [5],[7],[16]. A highlevel agent-based framework for protein structure prediction is present in [5]. The agents are separated on three layers, according to their knowledge and power. The first layer contains agents that explore the configuration space. A good coordination between these agents is considered to be necessary, this is achieved by some higher-level agents, whose role is to coordinate the search agents. The third layer is composed of cooperative agents, which implement a basic form of cooperation between lower-level agents. Another bioinformatics framework, called Evolution, is presented in [16]. This solution is based on a multi-agent system and uses a blackboard architecture. There are

three types of agents: model agents, algorithm agents and interface agents and the communication and coordination among them is achieved by a blackboard mechanism, which also allows data sharing. The blackboard is composed of several levels, each one recording the solution elements needed for the resolution of the problem. In [7] it proposing a distributed reinforcement learning based model for solving the bidimensional protein folding problem. The model proposed in [7] extends the reinforcement learning model that it has previously introduced for solving the problem. This model is based on a distributed Q-learning approach.

Other successful solutions consisted in transforming the protein structure prediction problem to a constraint minimisation problem with finite domain variables [2],[3]. This solution based on a search technique uses constraint programming to find the minimal energy for a given sequence of amino acids. This algorithm in [2],[3] is a combination of a Branch-and-Bound search together with a constrain-and-generate principle, as it is usual for constraint optimization.

NetLogo is regarded as one of the most complete and successful agent simulation platforms [20]. NetLogo is a high-level platform, providing a simple yet powerful programming language, built-in graphical interfaces and the necessary experiment visualisation tools for quick development of simulation user interface. It is a environment written entirely in Java, therefore it can be installed and activated on most of the important platforms.

In this paper we present an agent-based framework for protein structure prediction, composed by autonomous agents which collaborate in order to find a solution using Distributed Constraint Programming (DisCSP and DisCOP). Each amino acid of an input protein is viewed as an autonomous agent that communicates with others by transmitting messages. In this article was analysed the NetLogo environment [20] with the purpose of building a general model of implementation and simulation for the protein structure prediction [22]. This article synthesizes and extends all the tries of modelling and implementation in NetLogo for the protein structure prediction [14].

We provide a first working examples in open-source NetLogo, for this modelling. They can be downloaded from the websites [22]. Also, for this Netlogo models we have developed a methodology to run in a cluster computing environment. We utilize the Java API of NetLogo as well as LoadLeveler. LoadLeveler is a job scheduler written by IBM, to control scheduling of batch jobs [15].

2 The Protein Folding Problem

The Protein Structure Prediction Problem (PSP) is the problem of predicting the 2D/3D native conformation of a protein, when the sequence made of 20 kinds of amino acids (or residues) is known. The process for reaching the native state is known as the protein folding [1],[2],[3],[8],[9],[18].

For this PSP, there are several reduced computer modelling of proteins. In this paper we consider the ab initio modelling. These methods are based on the Anfinsen thermodynamic hypothesis [1]: the native conformation adopted by a protein is the one with minimum free energy.

Amino acids are modelled in a coarser way, usually as a single sphere. An important class of simplest models for proteins are lattice-based models - composed of a lattice that describes the possible positions of amino acids in space and an energy function of the protein, that depends on these positions. The goal is to find the global minimum of this energy function, as it is assumed that a protein in its native state has a minimum free energy and the process of folding is the minimization of this energy [1]. The simplifications used in this class of models are (1) monomers are represented using a unified size (2) bond length is unified (3) the positions of the monomers are restricted to positions in a regular lattice citeanfin. Thus, every conformation of a lattice protein is a self-avoiding walk in Z^2 or Z^3 (2D/3D dimensional lattice models). A discussion of lattice proteins can be found in [8],[9],[18]. However, the ab initio protein structure prediction problem has been shown to be NP-hard and a polynomial time algorithm is therefore unlikely to exist [4].

One of the most popular models of protein folding is the hydrophobic-hydrophilic (H-P) model in [8],[9]. In this model, the 20 letter alphabet of amino acids is reduced to a two letter alphabet, namely H(hydrophobic) and P(hydrophilic). The proteins are modelled as chains whose vertices are marked either H or P; the resulting chain is embedded in some lattice. An optimal embedding is one that maximizes the number of H-H contacts. Figure 1 shows a conformation for the sequence P H P P H H P H P P P P H H in the 2D square lattice whose energy is -3.

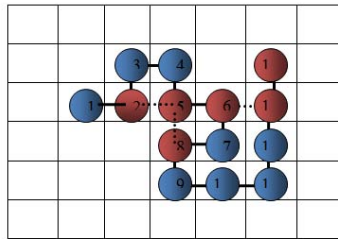


Fig. 1. Sample conformation for P H P P H H P H P P P P H H. The blue beads represent P, the red ones H monomers. The three contacts are indicated via dashed lines.

The primary structure of a protein P is seen as a sequence s of n amino acids (hydrophobic/ hydrophilic): $s = s_1, s_2, \dots, s_n$ where $s_i \in \{H,P\}, \forall 1 \leq i \leq n$.

Definition 1. (conformation). A conformation of the protein P is a function C, that maps the protein sequence s to the points of a 2D/3D dimensional Cartesian lattice: $C : [1..n] \rightarrow Z^d$. In fact, $s = s_1, s_2, \dots, s_n \rightarrow (x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ or $s = s_1, s_2, \dots, s_n \rightarrow (x_1, y_1, z_1), (x_2, y_2, z_2), \dots, (x_n, y_n, z_n)$ (where $d = 2$ or $d = 3$, $(x_i, y_i)/(x_i, y_i, z_i)$ - represents the position in the 2D/3D lattice of the amino acid s_i) such that:

1. $\forall 1 \leq i, j \leq n, \text{ with } i \neq j \implies c(i) \neq c(j)$ (self-avoiding).
2. $\forall 1 \leq i, j \leq n, \text{ with } |i - j| = 1 \implies |x_i - x_j| + |y_i - y_j| = 1$ (connected neighbours).

The first condition is the constraint that the conformation must be self-avoiding (the positions of two different amino acids must be different in the lattice).

The second condition is imposed by the lattice constraint (any two consecutive amino acids in the primary structure of the protein are neighbours in the 2D/3D lattice). It is considered that any position of an amino acid in the lattice may have a maximum number of 4 neighbours: up, down, left, right (2D lattice) or 6 neighbours: forward, backward, up, down, left, right (3D lattice).

Definition 2. (Conformation Energy). Given a sequence $s = s_1, s_2, \dots, s_n$, the energy of a conformation C of s is defined as follows:

$$- E(C) = \sum_{1 \leq i+1 < j \leq n} \varepsilon_{i,j} \cdot \delta(s_i, s_j)$$

where $\varepsilon_{i,j}$ is equal to -1 whenever both s_i and s_j are H amino acids, 0 otherwise, $\delta(s_i, s_j)$ is 1 if s_i and s_j are topological neighbours, 0 otherwise.

The energy function for the HP-model may be represented by the matrix which fixes the energy value of a contact between two monomers. Two residues are topological neighbours (TN), by being at the adjacent or immediate neighbouring lattice positions and not sequential in the original connectivity, as indicated by the dotted lines in figure 1. If two monomers of type H are topological neighbours, then an energy contribution (noted ε) is considered to have a value of -1, see [8],[9]. The sum of ε for HH interactions in a conformation using HP model is referred by E. Several different variations of the HP model, based on the values of the interaction potential, are shown in figure 2 (a),(b),(c) [8],[9],[11].

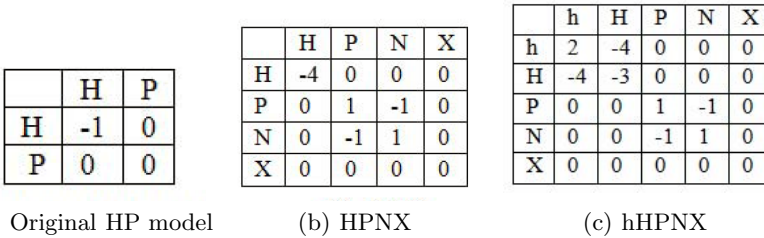


Fig. 2. Various interaction-potential matrices

The energy function in the HP model shows the fact that hydrophobic amino acids have a propensity to form a hydrophobic core. Any hydrophobic amino acid in a valid conformation C can have at most 2 such neighbours (except for the first and last amino acids, that can have at most 3 topological neighbours). Hence, a folding has minimum energy if it maximizes the number of HH contacts [1],[2],[3],[8],[9].

A solution for the 2D/3D HP protein folding problem, corresponding to a a sequence $s = s_1, s_2, \dots, s_n$, can be represented by a $n-1$ length sequence $L = dir_1, dir_2, \dots, dir_{n-1}$, where $dir_i \in \{L,R,U,D\}$ in 2D or $dir_i \in \{F, B, L, R, U, D\}$ in 3D, $1 \leq i \leq n-1$, where each position encodes the direction of the current amino acid relative to the previous one (F-forward, B-backward L-left, R-right, U-up, D-down) for the 2D square/3D cubic lattices.

3 DisCSP for Protein Folding Problem

3.1 The Distributed Constraint Satisfaction Problem

This paragraph presents some notions related to the DisCSP modelling [6],[12],[19],[21]. Constraint satisfaction is a classical and powerful tool in artificial intelligence whose traditional applications concern planning, scheduling, placement, logistics and so on. The Distributed Constraint Satisfaction Problem (DisCSP) has been formalized in [19].

Definition 3. *The model based on constraints CSP - Constraint Satisfaction Problem, existing for centralized architectures, is defined by a triple (X, D, C) , where: $X = \{x_1, \dots, x_n\}$ is a set of n variables; whose values are taken from finite domains $D = \{D_1, D_2, \dots, D_n\}$; C is a set of constraints declaring those combinations of values which are acceptable for variables.*

The solution of a CSP implies to find an association of values for all the variables that satisfy all the constraints.

Definition 4. *A problem of satisfying the distributed constraints (DisCSP) is a CSP, in which the variables and constraints are distributed among autonomous agents that communicate by exchanging messages. Formally, DisCSP is defined by a 5-tuple (X, D, C, A, ϕ) , where X, D and C are as before, $A = \{A_1, \dots, A_p\}$ is a set of p agents, and $\phi : X \rightarrow A$ is a function that maps each variable to its agent.*

Most of algorithms to solve DisCSP are distributed and asynchronous. To execute such asynchronous searching techniques, an agent has to be able to send messages to any other agents of its acquaintance set. For more details about DisCSP algorithms, see [6],[12],[19],[21].

3.2 The Distributed Constraint Optimization Problems

Distributed constraint optimization problems (DisCOP) represents a generic framework for the resolution of distributed problems in multi-agent systems (MAS), especially in problems where the challenge is to find the best value attribution for a set of variables that have interdependencies.

This paragraph presents some notions related to the DisCOP modelling [13],[17],[21]. DisCOP is a powerful tool for modelling a wide variety of multi-agent coordination problems such as distributed planning, distributed scheduling, distributed resource allocation and others.

Definition 5. *A Distributed Constraint Optimization Problem (DisCOP) is defined by a 6-tuple $(X, D, A, \phi, f_{i,j}, F)$ where X, D, A are as before, $A = \{A_1, \dots, A_p\}$ is a set of p agents, and $\phi : X \rightarrow A$ is a function that maps each variable to its agent, $f_{i,j}$ is a set of cost functions $f_{i,j} : D_i \times D_j \rightarrow \mathbb{N}$, to the pair of variables x_i and x_j , F is an objective function F , defined as an aggregation over the set of cost functions.*

The objective is to find the set of values A for the variables X , minimizing or maximizing the objective function. The function F is defined as $F(A) = \sum_{x_i, x_j \in X} f_{i,j}(d_i, d_j)$, where $x_i \leftarrow d_i$, $x_j \leftarrow d_j$ in A .

The cost functions in DisCOP are the analogue to the constraints from DisCSP and are sometimes referred to as valued or soft constraints. DisCOPs can be solved either using distributed search or by using distributed inference. About distributed search we mention the reference algorithm ADOPT [13] and their improved versions BnBADOPT. About distributed inference, we mention DPOP [17]. The Multi-Agent Systems from this paper is using the ADOPT algorithm.

ADOPT [13] is a backtracking based bound propagation mechanism. ADOPT was the first decentralized algorithm to guarantee optimality, while at the same time allowing the agents to operate asynchronously. The algorithm from [13] works as follows: first, the DFS structure is created. Then, backtrack search is performed top-down, using the following value ordering heuristic: at each point in time, each agent chooses the value with the smallest lower bound. It announces its descendants of its choice via VALUE messages, and waits for COST messages to come back from the children. Each agent adds the costs received from its children to the lower bound of the current value taken by the agent. If there is another value in the domain that has a smaller lower bound, the agent switches its value, and the process repeats, refining the lower bounds iteratively.

4 Modelling the Protein Folding Problem Using Distributed Constraints

In this section we will present two solutions of modelling for the protein folding problem using distributed constraints.

4.1 Modelling of the Protein Folding Problem in DisCSP

The previous presentation of the protein folding problem can be expressed as the following DisCSP:

- $A = \{A_1, \dots, A_n\}$ is a set of n agents that denotes a n amino acids exploring an environment, interacts and communicates with its spatial neighbours in order to minimize the energy function and sharing a few common information.
- $X = \{x_1, \dots, x_n\}$ is composed of variables storing the next heading (the next direction to explore of the current amino acid relative to the previous one) $x_i = (dir_i, px_i, py_i) / (dir_i, px_i, py_i, pz_i)$ of each amino acids of A_i where:
 - dir_i is the next direction;
 - $(px_i, py_i) \in Z^2$ is the position, $px_i, py_i \in [-n, n]$, (respectively in 3D, $(px_i, py_i, pz_i) \in Z^3$, $pz_i \in [-n, n]$).
- $D = \{\text{dom}(x_1), \text{dom}(x_2), \dots, \text{dom}(x_n)\}$ with $\text{dom}(x_i)$ ($1 \leq i \leq n$) is the set of all 4/6 cardinal directions that a amino acids A_i can choose to plan its next movement (for the 2D square, 3D cubic lattices and 2D triangular lattices).

- $C = C_1 \cup C_2$ where:
 - $C_1 = \forall A_i \in A, \forall A_j \in A, \text{ with } i \neq j \implies (px_i, py_i) \neq (px_j, py_j)$ (self-avoiding).
 - $C_2 = \forall A_i \in A, \forall A_j \in A, \text{ with } |i - j|=1 \implies |px_i - px_j| + |py_i - py_j| = 1$ (connected neighbours).

To find a conformation, the amino acids of set A have to periodically solve this DisCSP in order to be able to choose a direction compatible with the requirements previously introduced and in order to minimize the energy function. The algorithm DisCSP are iterated until a valid conformation is found. If we have found a valid conformation, then the energy of a conformation c is determined *CurrentValEnergy(c)*. Then the additional constraint *CurrentValEnergy(c) < MinValEnergy* is added, and the search is continued in order to find the next conformation, which must have a smaller energy.

4.2 Modelling of the Protein Folding Problem in DisCOP

The solution for DisCOP modelling is similar to the one for DisCSP. According to the definition presented in the previous paragraph, a DisCOP modelling is defined by a 6-tuple $(X, D, A, \phi, f_{i,j}, F)$ where X, D, A are as before (A is a set of n agents that denotes a n amino acids, X is composed of variables storing the next heading and D is the set of all 4/6 cardinal directions). The previous presentation of the protein folding problem requires in addition:

- Hard constraints: $C_h = C_{h1} \cup C_{h2}$, which impose that the conformation must be self-avoiding (the positions of two different amino acids must be different in the lattice) and connected neighbours (any two consecutive amino acids in the primary structure of the protein are neighbors in the 2D/3D lattice), where:
 - $C_{h1} = \forall A_i \in A, \forall A_j \in A, \text{ with } i \neq j \implies (px_i, py_i) \neq (px_j, py_j)$ (self-avoiding).
 - $C_{h2} = \forall A_i \in A, \forall A_j \in A, \text{ with } |i - j|=1 \implies |px_i - px_j| + |py_i - py_j| = 1$ (connected neighbours).
- Soft Constraint : $C_s = \{f_{i,j} : D_i \times D_j \longrightarrow R, \forall 1 \leq i,j \leq n, \text{ with } i \neq j\}$ is based on the energy function for the HP-model (where a cost of infinity indicates a forbidden tuple by the hard constraints). The cost functions $f_{i,j}$ may be represented by the matrix which fixes the energy value of a contact between two monomers A_i and A_j (fig 3.a,b).
- The function F is similar to the energy of a conformation C:
 - $F(C) = E(C) = \sum_{1 \leq i+1 < j \leq n} f_{i,j}$

The goal of the optimization is to find the conformation C which (a) is feasible (i.e. respects all constraints) and (b) minimizes the sum of the agents' utilities (the minimization of this energy E).

Example: Consider an example where 7 agents (amino acids) want to find the optimal conformation in order to minimize the energy function. Each agent A_i has a local problem composed of:

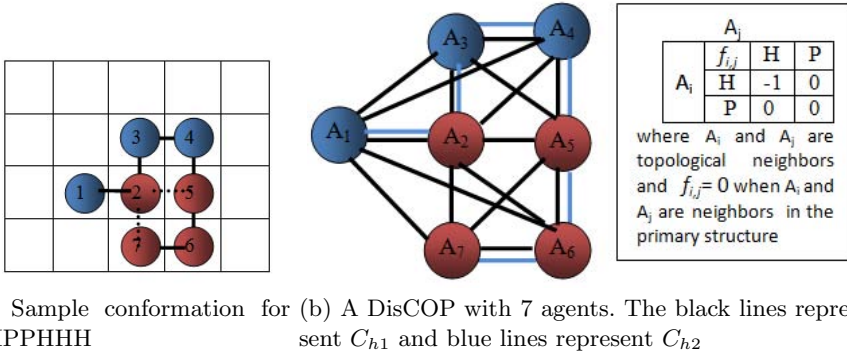


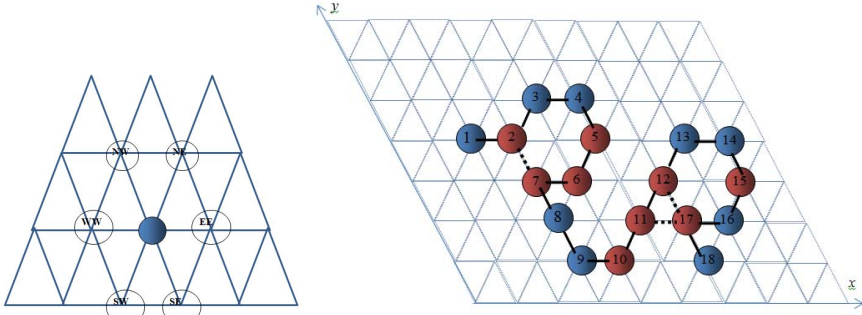
Fig. 3. A PSP example in DisCOP model

- variable $x_i = (dir_i, px_i, py_i)$ or $x_i = (dir_i, px_i, py_i, pz_i)$;
- domains: the set of all 4/6 cardinal directions.
- hard constraints which impose that the conformation must be self-avoiding and connected neighbours;
- utility functions: the matrix which fixes the energy value of a contact between two monomers A_i and A_j ; In this cost function he have associated the infinity value to the combination in which two agents A_i and A_j have the same coordinates in the lattice. Also, the cost function takes the infinity value if the two agents, consecutive amino acids in the primary structure of the protein are not neighbours in the 2D/3D lattice. For example, $f_{i,j} = \infty, \forall A_i, A_j \in A$ with $(px_i, py_i) = (px_j, py_j)$

4.3 Lattice Models

Lattice models have long been used for protein structure prediction and representation of 2D/3D structures. A discussion of lattice models can be found in [18], [10]. The discretization of the conformational space is necessary for modelling and analysing the computational complexity of PSP problems. The lattice model determines the space of possible conformations for a given protein. Thus, there are several lattice models [18], [10]. The 2D square lattice and the 3D cubic lattice are the most thoroughly studied lattices. The multi-agent system from this paper is using the well-known lattice bead models (2D square, 3D cubic and 2D/3D triangular lattice).

In the two models proposed in DisCSP and DisCOP, each amino acid is considered as an agent A_i and owns a variable to instantiate: MyValue = (dir_i, px_i, py_i) or MyValue = $(dir_i, px_i, py_i, pz_i)$. This variable represents its next heading (the next direction to explore of the current amino acid relative to the previous one) and position of amino acids A_i . The domain of each variable in 2D square/3D cubic lattices is composed with the 4/6 cardinal directions: (F-forward, B-backward L-left, R-right, U-up, D-down). For the 2D triangular lattices the domain of each variable is composed with the 6 cardinal directions:(NW, NE, WW, SW, SE,



(a) The 2D triangular lattices (b) Sample conformation for PHPPHHHPHPPHPPH-PPH. The three contacts are indicated via dashed lines.

Fig. 4. The two-dimensional triangular lattice for the protein folding problem-2D HP

EE). The 2D triangular lattices are shown in fig 4.a. Figure 4.b. shows a conformation for the sequence PHPPHHHPHPPHPPH whose energy is -3.

5 Multi-agent System for Modelling and Simulation of the Protein Folding Problem in NetLogo

In this section we present the way of implementation of the multi-agent systems for the protein folding problem in Netlogo, using three crystal lattice models. We provide two solutions in open-source NetLogo, for implementation and simulation of the protein structure prediction with ABT algorithm and ADOPT algorithm. The two versions of the multi-agent system can be found on the site in [22].

In figure 5 is presented this multi-agent system’s architecture for modelling and simulation of the protein folding problem in NetLogo. The generic framework architecture will be structured on two levels, corresponding to the two stages of implementation [21]. The first level refers to the way of representing the surface of the multi-agent system. This is the exterior level. The definition of the way in which asynchronous techniques will be programmed so that the agents will run concurrently and asynchronously will be the internal level of the model. To this level are simulated the agents associated to the amino acids and are implemented the DisCSP/DisCOP algorithms for finding an optimal conformation.

The features of each sublevel of the multi agent system are described below:

- HP sequence level. On this level are selected HP sequences, which can be either generated by the NetLogo module *HP generator* or loaded from a file (introduced by the user in a file). An HP sequence is represented as a chain of n characters belonging to a same length two-element alphabet {H, P};
- Initial Conformational space level. This level corresponds to the initial conformational space created when the HP model, being discrete or continuous,

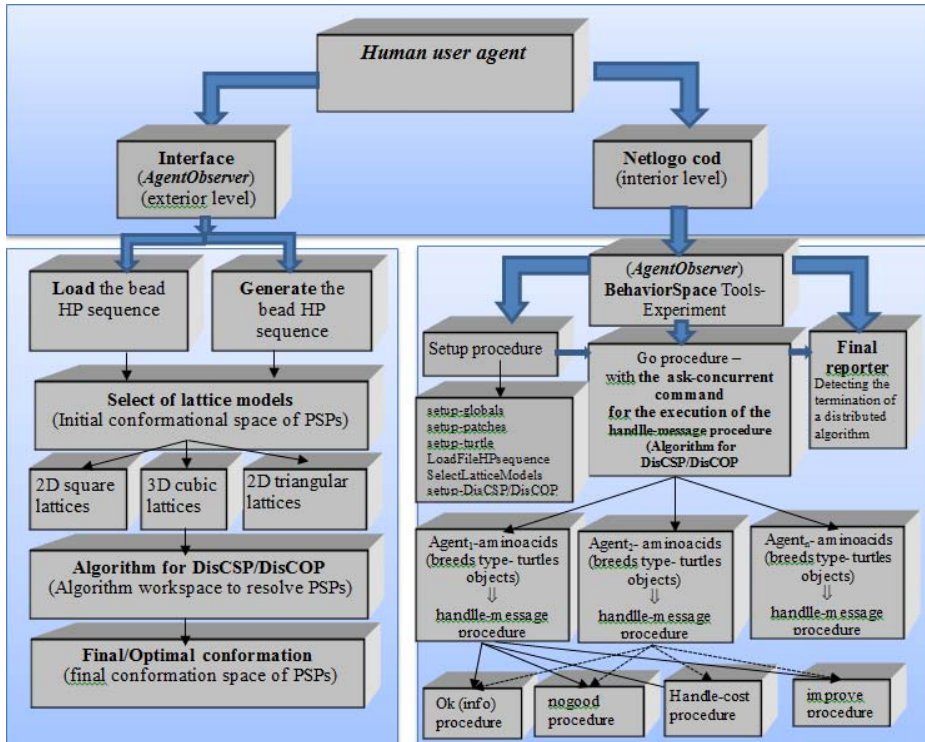


Fig. 5. Architecture of a multi-agent system for modelling and simulation of PSPs in NetLogo

is executed on 2D triangular, 2D square or 3D cubic lattices. At this level, the human user selects lattice models for 2D/3D conformation from the 3 available. However, the framework can be adapted, with minor changes, to handle other types of discrete lattices.

- Algorithm workspace level. On this level is selected the DisCSP/DisCOP algorithm for finding an optimal conformation.

As regarding the internal level, the multi-agent system needs to simulate the agents of amino acids type. First of all, the agents are represented by the breed type objects (those are of the turtles type). Fig. 6 shows the way the agents are defined together with the global data structures proprietary to the agents.

For building of the user interface that contains various DisCSP/DisCOP problem's objects (amino acids, bonds, etc.) are used objects of the patch type.

Concretely, for the case of the protein folding problem the representation of amino acids, bonds and lattice is modelled as a grid with a number of cells. Each cell can be empty or occupied by a amino acid. To model the surface of the application are used objects of the patches type. In fig. 7 and fig. 8 is captured an implementation of the ABT technique for the protein folding problem (2D square lattice and 3D cubic lattice) that uses the model presented.

```

breeds [agents-aminoacids bondes]
globals[variables that simulate the memory shared by all the agents:
; NrStep MinValEnergy CurrentValEnergy ]
agent-amino acids-own [message-queue nogoods MyValue MyContext ValEnergy ...]
;message-queue contains the received messages.
;[[dir0 px py] [dir1 px py] ...] diri = direction of the current
amino acid relative to the previous one, [diri px py]= -1 if unknown.
;nogoods is the list of inconsistent positions [0 1 1...] where 1 is inconsistent
    
```

Fig. 6. Agents' definition in DisCSP-Netlogo

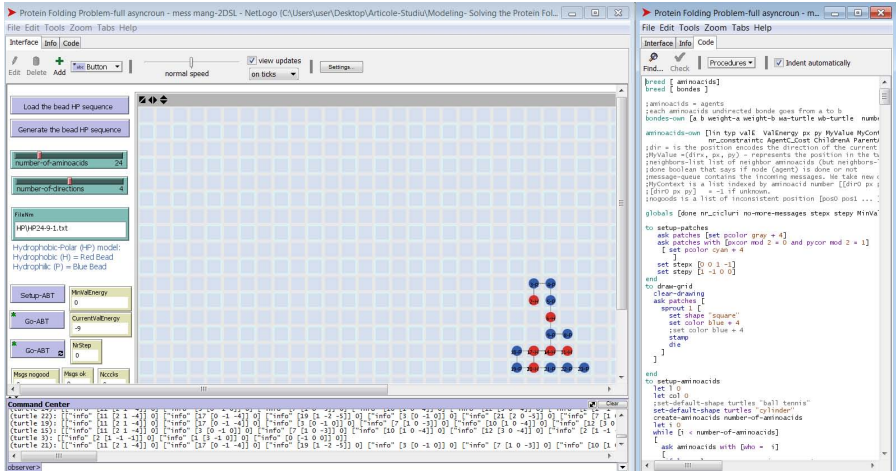


Fig. 7. NetLogo implementation for the protein folding problem-2D square HP

In [15] we presented a methodology to run the NetLogo models for the asynchronous searching techniques (DisCSP/DisCOP) in a cluster computing environment or on a single machine. The solution from [15] uses the Java API of NetLogo as well as LoadLeveler. LoadLeveler is a job scheduler written by IBM, to control scheduling of batch jobs. This solution can be used on any cluster with Java support. Such a solution will allow running a large number of agents (amino acids). The first experiments were done on the InfraGrid cluster from the UVT HPC Centre [23], on 100 computing systems.

The proposed solution in [15], runnable without the GUI on a single computer or on a cluster, is used a tool named BehaviourSpace, existent in NetLogo. BehaviorSpace is a software tool integrated with NetLogo that allows you to perform experiments with models in the "headless" mode, that is, from the command line, without any graphical user interface (GUI).

The solution from [15] supposes modifications at the internal level of the multi-agent system specifically defining special procedures to run the code for the DisCOP/DisCSP algorithms, according with the methodology from [15]. Also, running the multi-agent system supposes the creation of an experiment

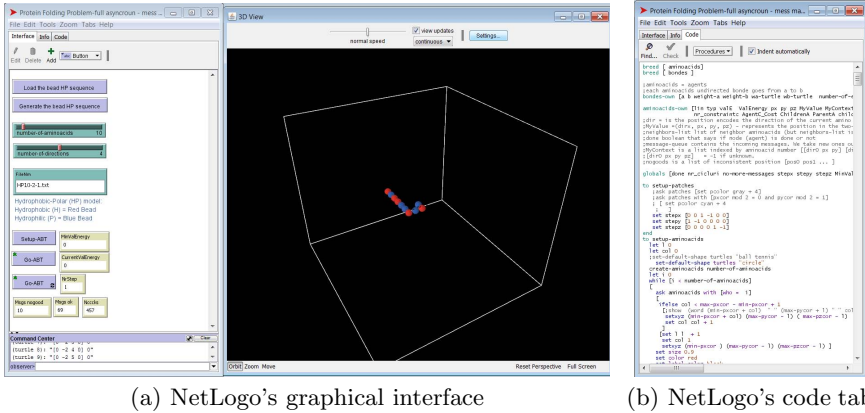


Fig. 8. NetLogo implementation for the protein folding problem-3D cubic HP

using BehaviorSpace and parse the NetLogo file into an input XML file (so that it can be runned in the headless mode, that is without GUI).

6 Conclusion

In this paper we present multi-agent systems for protein structure prediction, composed by autonomous agents which collaborate in order to find a solution using Distributed Constraint programming (DisCSP/DisCOP). Each amino acid of an input protein is viewed as an autonomous agent that communicates with others by transmitting messages in order to minimize the energy function. Starting from the proposed implementation models with constraints, in this article we present multi-agent systems that can be used for the implementation and simulation of the protein structure prediction, that can run on a single computer or on a cluster computing environment.

This framework was designed as a virtual laboratory to be used for the study of the protein folding, initially using the well-known lattice bead models (2D square, 3D cubic and 2D/3D triangular lattice). We plan to extend the multi-agent systems from this paper so that it will use other DisCOP algorithms such as DPOP or BnBADOPT algorithms in both variants: single variable and multiple local variables. Also, we wish to extend the evaluation of asynchronous searching techniques in this new model, for some large HP protein sequences, to further test its performance.

Acknowledgments. This work was partially supported by the European Commission grant FP7-REGPOT-CT-2011-284595 (HOST) and by the Romanian national grant PN-II-ID-PCE-2011-3-0260 (AMICAS).

References

1. Anfinsen, C.B.: Principles that govern the folding of protein chains. *Science* 181, 223–230 (1973)
2. Backofen, R., Will, S., Bornberg-Bauer, E.: Application of constraint programming techniques for structure prediction of lattice proteins with extended alphabets. *Bioinformatics* 15, 234–242 (1999)
3. Backofen, R.: The protein structure prediction problem: A constraint optimization approach using a new lower bound. *Constraints* 6(2-3), 223–255 (2001)
4. Berger, B., Leighton, T.: Protein folding in HP model is NP-complete. *Journal of Computational Biology* 5, 27–40 (1998)
5. Bortolussi, L., Dovier, A., Fogolari, F.: Agent-based protein structure prediction. *Agent-based protein structure prediction. Multiagent and Grid Systems* 3(2), 183–197 (2007)
6. Bessiere, C., Brito, I., Maestre, A., Meseguer, P.: Asynchronous Backtracking without Adding Links: A New Member in the ABT Family. *Artificial Intelligence* 161, 7–24 (2005)
7. Czibula, G., Bocicor, M.I., Czibula, I.G.: Solving the Protein Folding Problem Using a Distributed Q-Learning Approach. *International Journal of Computer* 5(3), 404–413 (2011)
8. Dill, K., Lau, K.: A lattice statistical mechanics model of the conformational sequence spaces of proteins. *Macromolecules* 22, 3986–3997 (1989)
9. Dill, K.A., Bromberg, S., Yue, K., Fiebig, K.M., Yee, D.P., Thomas, P.D., Chan, H.S.: Principles of protein folding - a perspective from simple exact models. *Protein Science* 4(4), 561–602 (1995)
10. Hart, W., Newman, A.: Protein Structure Prediction with Lattice Models. In: *Handbook of Computational Molecular Biology*. Chapman & Hall CRC Computer and Information Science Series, pp. 30-1–30-24 (2006)
11. Hoque, T.M., Chetty, M., Sattar, A.: Extended HP Model for Protein Structure Prediction. *Journal of Computational Biology* 16(1), 85–103 (2009)
12. Meisels, A.: *Distributed Search by Constrained Agents: algorithms, performance, communication*, pp. 105–120. Springer, London (2008)
13. Modi, P., Shen, W.-M., Tambe, M., Yokoo, M.: ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence* 161(1-2), 149–180 (2005)
14. Muscalagiu, I., Iordan, A., Osaci, M., Panoiu, M.: Modelling and simulation of the protein folding problem in DisCSP-Netlogo. *AWERProcedia Information Technology and Computer Science* 2, 137–144 (2012)
15. Muscalagiu, I., Popa, H.E., Vidal, J.: Clustered Computing with NetLogo for the evaluation of asynchronous search techniques. In: *12th International Conference on Intelligent Software Methodologies, Tools and Techniques*, Budapest, Hungary (2013)
16. Perez, P.P.G., Beltran, H.I., Rojo-Dominguez, A.: Multi-agent systems applied in the modelling and simulation of biological problems: A case study in protein folding, pp. 128–138. *World Academy of Science, Engineering and Technology* (2009)
17. Petcu, A.: *A Class of Algorithms for Distributed Constraint Optimization*. PhD. Thesis No. 3942, Swiss Federal Institute of Technology (EPFL), Lausanne (2007)
18. Skolnick, J., Kolinski, A.: Reduced models of proteins and their applications. *Polymer* 45, 511–524 (2004)

19. Yokoo, M., Durfee, E.H., Ishida, T., Kuwabara, K.: The distributed constraint satisfaction problem: formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering* 10(5), 673–685 (1998)
20. Wilensky, U.: *NetLogo itself: NetLogo*. Center for Connected Learning and Computer-Based Modelling. Northwestern University, Evanston (1999), <http://ccl.northwestern.edu/netlogo/>
21. Vidal, J.: *Fundamentals of Multiagent Systems with NetLogo Examples* (2009), <http://multiagent.com/2010/02/multiagent-systems-textbook.html>
22. Muscalagiu, I.: *MAS NetLogo Models*, <http://discsp-netlogo.fih.upt.ro/>
23. *InfraGRID Cluster*, <http://hpc.uvt.ro/infrastructure/infragrid/>

Modeling AIDS Spread in Social Networks An In-Silico Study Using Exploratory Agent-Based Modeling

Muaz A. Niazi^{1,*}, Amnah Siddiqua², and Giancarlo Fortino³

¹ Bahria University, Islamabad, Pakistan
muaz.niazi@ieee.org

² Atta-ur-Rahman School of Applied Biosciences, Islamabad, Pakistan
amnahsiddiqua@gmail.com

³ University of Calabria, Italy
giancarlo.fortino@unical.it

Abstract. The Acquired Immunodeficiency Syndrome (AIDS) epidemic has perhaps one of the most complex social structures exhibited by any epidemic. AIDS spread is strongly linked with social networks transgressing cultural, religious and geographical boundaries making it difficult to conduct an objective study. Agent-based Modeling is well-known as an effective method for the exploration and study of complex systems. In this paper, using an example from three different types of populations, we present an Exploratory Agent-based Model (EABM) of AIDS in hybrid populations. Calibrated with data from UNAIDS studies, the model demonstrates how modeling using EABM can be useful to study the complexity in complex social systems in the absence of data of complex interactions. Extensive simulation experiments demonstrate the suitability of this proposed approach to study complex social phenomena.

Keywords: Agent based modeling, AIDS, Complex Adaptive System, sub-populations, complex networks, Cognitive agent-based computing.

1 Introduction

The complexity in AIDS has origins from its medicinal complexity with the RNA retrovirus with one of the most complex known interactions with human DNA [1] to social dynamics associated with its mode of spread [2]. One particularly complex problem is the study of concurrent [3] and multiple partnerships [4] in AIDS. In Pakistan, the problem of an increase of the infection has been noted recently due to both injecting as well as commercial sexual networks [5]. Alternatively there have been some previous studies examining the network structure of AIDS, they are often limited by their focus on either a single population [6] or else on a particular group such as high-risk groups [7]. While there have been previous studies modeling AIDS as a Complex Social Network, such models cannot always be readily developed for cultures where religious and moral values

* Corresponding author.

“expect” individuals to “never” be infected [8]. While it is important to be able to model various structures and dynamics of the AIDS based Complex Adaptive Systems (CAS) [9], the absence of effective tools to model complexities of the epidemic from the social network perspective poses a set of difficulties.

Agent-based modeling (ABM) offers a set of computational methods allowing for the study of and experimentation with various CAS [10]. In this paper, we develop an Exploratory Agent-based Model (EABM) based on the Cognitive Agent-based Computing Framework presented earlier by Niazi and Hussain in [11]. The proposed EABM is a model of an artificial society with the goal of giving the ability to study the effects of morality and commitment of male population with their (female) partners coupled with the use of effective protection mechanisms (such as condoms) on the spread of AIDS/HIV in different populations. The model allows the study of the AIDS epidemic spread in social networks. Our results of extensive simulation experiments calibrated using healthcare data from UNAIDS (the Joint United Nations Programme on HIV/AIDS) country reports for high risk population groups, we demonstrate how the model can be used to study trends and complexities in AIDS spread.

The structure of the rest of the paper is as follows: Section 2 presents the background and related work. In section 3, we describe an exploratory agent-based model [12] used to perform this in-silico investigation. This is followed by a detailed discussion of experimental results in section 4 which have been calibrated using data made available by the United Nations UNAIDS/WHO [13], followed by conclusions.

2 Background and Related Work

The Acquired Immunodeficiency Syndrome (AIDS) epidemic can be considered as an emergent outcome based on the numerous interactions of a large number of heterogeneous entities or agents. The HIV/AIDS epidemic has received a considerable interest since the early 80's [14]. However, in spite of this considerable amount of literature and wide-spread knowledge, the AIDS epidemic still poses a serious problem with world-wide prevalence. This section provides the necessary background and related work about agent-based modeling of HIV/AIDS spread.

2.1 Exploratory Agent-Based Modeling (EABM)

Agent based modeling (ABM) is a simulation paradigm which has close ties with actual scientific experiments thus making it a good technique for evaluation of different paradigms concepts [15]. In the Cognitive Agent-based Computing framework, agent-based and complex network-based models are structured in a hierarchy of 4 different levels. Level 2 of the framework is correlated with existing literature in ABM in the form of EABM. The goal of this modeling level is experiment and develop proof-of-concept models of CAS with the goal of performing experimentation for improving our understanding about a particular real-world complex system. Previous work has demonstrated the use of

Exploratory ABM (EABM) in different multidisciplinary studies as diverse as for the simulation of research [16] to modeling breast cancer [17]. However, unlike Validated agent-based modeling [18], this model is more for experimentation rather than correlation with exact dynamics which can be performed with more in-depth studies using Validated Agent-based Modeling by developing a Virtual Overlay Multiagent System (VOMAS).

While there have been some studies which have demonstrated the use of ABM for the study of AIDS/HIV such as [19] and [20], the current paper has the following points which are not previously covered in these other works:

1. We examine AIDS spread across high risk as well as low-risk groups.
2. Our proposed ABM allows for a study of the flow of disease from one population to another and then back.
3. The model uses calibration data from the UNAIDS country report to realistically model the particular setting in Pakistan, a country for which there has not been any available HIV/AIDS model.

3 Methodology

In this study, we present the design of an agent-based model for the evaluation of AIDS as an epidemic spreading across hybrid population sub-groups in Pakistan. The model has been designed as a representative of a hybrid population consisting of various sub-populations that were categorized on the basis of risk behavior. Unlike social network approaches, which consider ties between individuals as static, the ABM demonstrates a more realistic scenario by portraying the dynamics in networks across populations. These included one HIV/AIDS high risk population; i.e., Female Sex Workers (FSWs)¹ and three seemingly low risk populations: 1) heterosexual men (termed “primaries”) being the people who have occasional interaction with the FSWs as their clients), 2) heterosexual women (termed “secondaries”) since they do not have any direct interaction with the FSWs and acquire infection via primaries being their partner), and 3) heterosexual women who are not committed partners of the primaries, termed as exsecondaries.

This section describes the various entities of the agent-based model of HIV spread developed using NetLogo [21], a freely available agent-based modeling tool that has previously been used extensively to model complex adaptive systems.

3.1 Model Design

The ABM is made up of three populations sub-groups: FSWs, their clients (termed as primaries), and the partners of the primaries (termed as secondaries)

¹ We would like to note here that while FSWs were chosen in this particular model, they have been selected purely for the purpose of demonstration of the ABM technique and not as a representative of being the only high risk population involved in the spread of AIDS in a social network.

as well as extra secondaries (described later in Model Input Parameters) as can be noted in Fig. 1a. To differentiate between them, a color code has been used where yellow is used for the FSWs, blue for the primaries and pink for the secondaries. The color red has been used to indicate HIV/AIDS infected individuals.

Agent-Interactions. The model used “links” to represent two people engaged in an intimate relationship. These links created couples. There were essentially two kinds of couplings envisioned in this model:

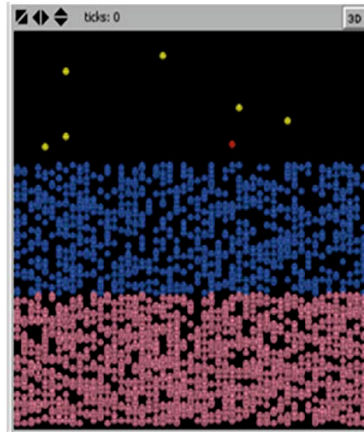
1. An interaction between FSWs and primaries
2. An interaction between primaries and secondaries. These again were of two types:
 - (a) Interactions between primaries and their socially selected secondary partners (e.g. married partner)
 - (b) Interactions of primaries with individuals other than FSWs and secondaries

To perform verification of the model, each time there was a coupling, a link was shown on the screen between the two agents as a connecting line as can be observed in Fig. 1b.

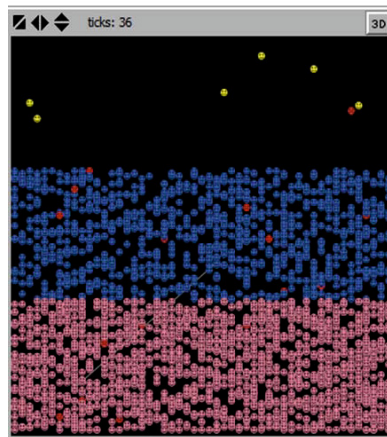
Model Input Parameters for Modification of Agent Behavior. Several input parameters were provided as controls in the model to adjust the behavior of the agents. The number of sub-populations could be adjusted through the sliders on the graphical user interface (GUI) of the model as can be noted in Fig. 2. The differentiation between secondaries of the two types can also be adjusted with the help of another slider named “ex-sec”, where ex-sec depicts the low risk heterosexual women other than the formal (life) partners of the primaries. The coupling between primaries and secondaries was controlled through a coupling variable based on an average monthly frequency of intimate interactions (coupling) and the coupling between the primaries and FSWs was controlled through a variable adjusted according to the average client coupling recorded for the FSWs. Two key confounding parameters; i.e. the commitment level of the primaries and the condom-usage among the primaries was used to study their impact on the spread of HIV/AIDS. The values of these parameters could also be adjusted according to the sliders provided in the model.

Model Output Parameters. The model has been designed such that changes occur over time. In other words, outputs of the model are time-dependent. We can write this as

$$O(t + 1) = F(O(t), I) \quad (1)$$



(a) Screen shot of agents used during simulation modeling showing female sex workers, FSWs (yellow), infected FSWs (red), their clients, termed primaries (blue) and partners of primaries, termed secondaries (red)



(b) Screen shot of sexual coupling between agents of different sub-populations during simulation as shown by vertical lines between agents. For each time tick (equally-spaced time interval), the interactions appear as lines between the relevant agents and then disappear in the next time tick for the sake of clarity

Fig. 1. Simulation using the NetLogo Agent-based Modeling tool

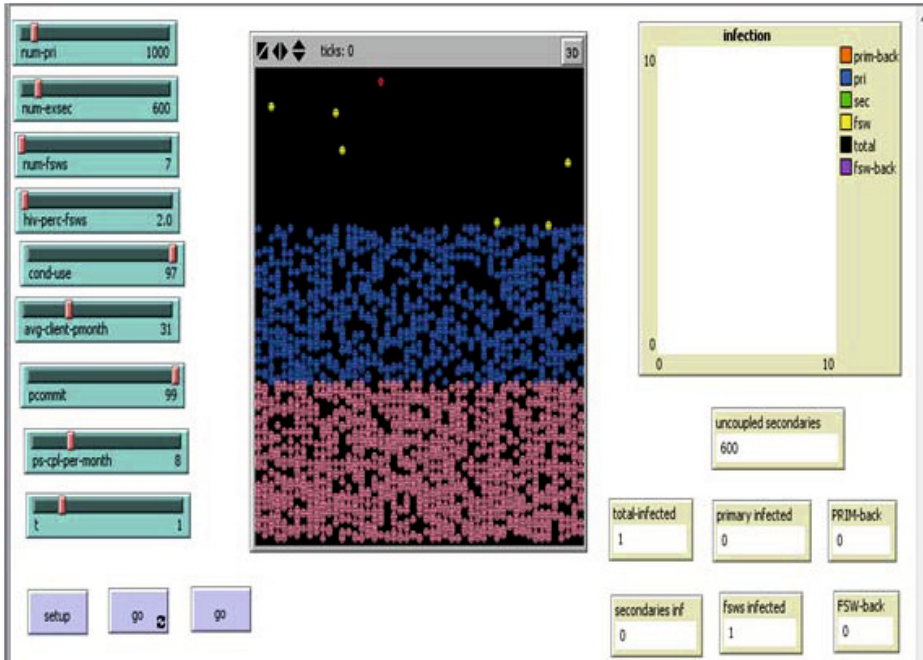


Fig. 2. Screen shot of the graphical user interface (GUI) of the agent-based model of the HIV spread in FSWs through back inflow. The image shows sliders that allowed modulation of various sub-populations within the model as well as the level of confounding parameters like commitment levels and condom usage among the various agents.

Where O stands for the aggregate output of the model. (Thus while the model itself is stochastic, in general, the aggregate output is deterministic)

And F depicts a function such that the aggregate output of the model can be found using the previous outputs as well as the input variables I .

Several output monitors have been designed into the model. In addition, plots and counters are used for displaying the state variables of various agents at any given time. Outputs include estimates of the number of infected FSWs, their direct partners and secondaries. Furthermore, the model also provides an estimate of the number of non-committed-secondaries (the number of secondaries used in the simulation which were not committed with any of the primaries), non-committed-infected-secondaries (the number of infected non-committed secondaries), and the total-infected population, a number that gives the overall picture of the infection spreading in the population. More importantly, the model also displays the numbers of FSWs which are infected by means of an infection transferred on to clients, their partners and back to the FSWs (FSW-back-infected). In other words, these infections resulted due to the interaction of infected primaries with non-infected FSWs, a value that demonstrates the effects of the existence of a complex hidden network for HIV back-flow. Finally, the number

of back-infections of primaries from secondaries; i.e., primaries-back-infected can also displayed individually. The infection curve for all sub-populations can be examined using a line plot.

Model Calibration. To ensure realism in the simulation output, the input data was calibrated using realistic data. The calibrated parameters included the agent population, the coupling parameter of primaries and FSWs, condom usage, infection rate, and time period. The only parameter that was not based on real data was the coupling parameter between primaries and secondaries. The idea in this design is to be able to examine different model worlds.

4 Results and Discussion

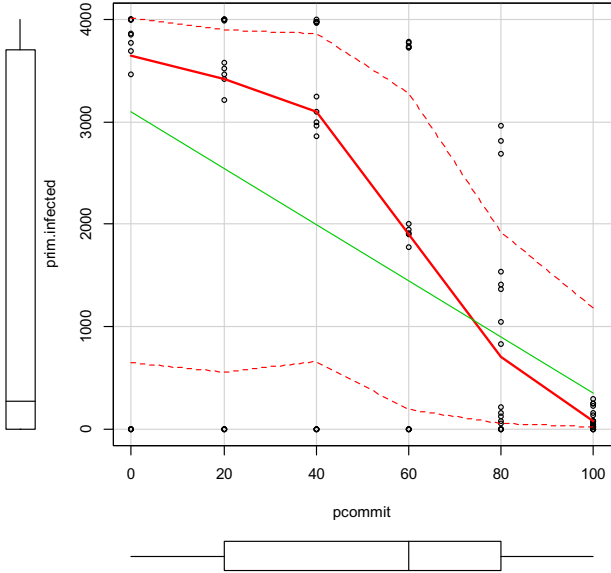
Simulations were calibrated using the 2003 and 2005 HIV reports from the Pakistan National AIDS Control Program (NACP) [13]. According to the report, condom usage was 25% and the ratio of FSWs to male population was 7:1000 (FSW prevalence rate in Pakistan). An error line function was used to get a clear picture of the minimum, maximum and average values of the data (to a 95% confidence interval) to observe various trends as can be noted in Fig. 3. Each point in the graphs was a result of 50 individual simulations, thereby minimizing the effects of random variables. While our goal was to develop an exploratory model of Pakistan population, the results depict interesting effects and possible trends which may be examined by empirical data collection exercises.

4.1 Discussion

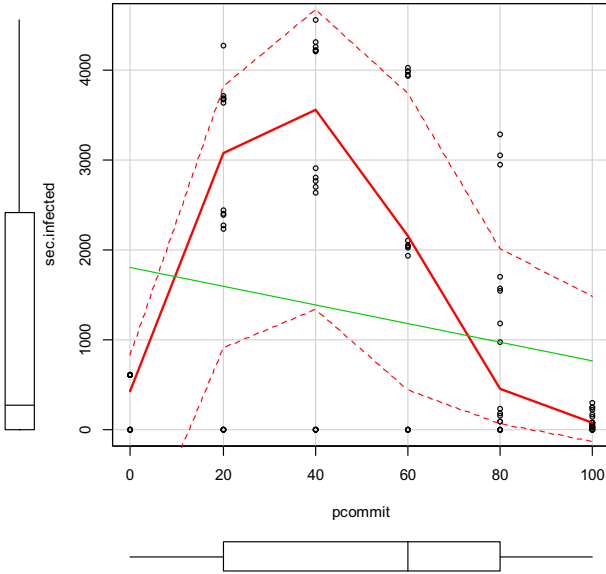
While the goal of this paper is to present the model itself, for the sake of validation and demonstration, we studied one key factor of using simulation experiments. The primary experiment conducted was based on a variation in the commitment level of the primaries with details as follows:

The frequencies of coupling as well as all other parameters were calibrated in the same ratio as specified by the NACP/UNAIDS reports. Each simulation was repeated with ($n = 50$) for each set of parameters to remove stochastic effects. Every clock tick of simulation time represented one month. Simulation experiments were developed to test the effect of commitment level of the primary partners of FSWs on the spread of infection among the various agents being tested. The study was limited to the commitment level of the primary population since intuitively, it was the actions of this populations that forms and shape the spread of the HIV epidemic within the chosen population model.

Here, we vary the commitment level in primaries and notice the effects of the infections in primaries as well as secondaries. Firstly, we note that when the infections in the primaries decreases if the commitment level is increased. Thus the more committed the primaries are with their partners, the lesser chance of spread of infections inside this population. This fact is also understandable from the perspective of actual facts. Now, in this particular model, at the start,



(a) Infected Primaries



(b) Secondaries Infected

Fig. 3. Results of simulation modeling of infection spread by varying the commitment level among primaries. The graphs represent the infected agents with increase in commitment levels.

the infection is only evaluated in the FSW population. However the model is dynamics so while initial infections are due to FSWs, future infection dynamics are considerable complex as can be seen in figure 3a.

Next, let us examine what happens in terms of infections in the secondaries. We notice in figure 3b that the results actually follow close to a Bell curve. In other words, the results are stochastic in nature. The results are somewhat surprising and emergent in nature because for very low and very high commitment levels, the results of infections are very low. However, with an average commitment level, the infections are spreading more. In other words, further investigation is needed as to why this type of infection behavior is emerging in the population. However, this also validates the simulation because the secondaries infected less at low commitment level might also be demonstrating the fact that people with low commitment level would couple less often with their original partners going more often to their other partners like FSWs as well as other partners.

5 Conclusions and Future Work

In this paper, we have presented an exploratory agent-based model for studying various dimensions of complexity in an artificial society consisting of various sub-populations. The goal of this model is to allow for the development of exploratory experiments which can be correlated with models and results from the real world. Using UNAIDS data, we have demonstrated the effectiveness of the proposed approach of studying the dynamics of AIDS. Interesting results from simulation experiments include non-linear and unexpected emergent patterns in populations based on relation changes with different segments of population.

While, it may appear easy to discard the outcomes demonstrated by the results from a simulation model, we would like to mention that these simulation experiments were repeated numerous times. In addition, results were generated using parameter-sweeping. It is such methods which is why Axelrod has noted that ABM is a “third way of doing science” and has the goal of “aiding intuition”[22]. As such, these results can actually predict considerable expected patterns in populations as well as pave way for conducting studies across populations. Thus, we note that ABM in general, and Exploratory Agent-based Modeling in particular may assist researchers in developing cost-effective preventive strategies for the AIDS epidemic. In the future, the model can be extended and enhanced to support other dynamics of AIDS in populations. We have also identified the need for further investigation of AIDS spread because the model is generating emergent behavior which needs further exploration perhaps by means of survey or network studies. Possible extensions of the framework include the use of statechart-based formalism such as presented earlier by Fortino et al. in [23] and other techniques discussed by Cossentio et al. in [24].

References

1. Barnes, D.M.: Aids research in new phase: The complexity of the aids virus. *Science* 233(4761), 282–282 (1986)
2. Bekele, T., Rourke, S., Tucker, R., Greene, S., Sobota, M., Koornstra, J., Monette, L., Rueda, S., Bacon, J., Watson, J., et al.: Direct and indirect effects of perceived social support on health-related quality of life in persons living with hiv/aids. *AIDS Care Psychological and Socio-medical Aspects of AIDS/HIV* (2012)
3. Morris, M., Kretzschmar, M.: Concurrent partnerships and the spread of hiv. *Aids* 11(5), 641–648 (1997)
4. Johnson, A.M., Mercer, C.H., Erens, B., Copas, A.J., McManus, S., Wellings, K., Fenton, K.A., Korovessis, C., Macdowall, W., Kiran, Nanchahal, o.: Sexual behaviour in britain: partnerships, practices, and hiv risk behaviours. *The Lancet* 358(9296), 1835–1842 (2001)
5. Bokhari, A., Nizamani, N.M., Jackson, D.J., Rehan, N.E., Rahman, M., Muzaffar, R., Mansoor, S., Raza, H., Qayum, K., Girault, P., et al.: Hiv risk in karachi and lahore, pakistan: an emerging epidemic in injecting and commercial sex networks. *International Journal of STD & AIDS* 18(7), 486–492 (2007)
6. Klovdahl, A.S., Potterat, J.J., Woodhouse, D.E., Muth, J.B., Muth, S.Q., Darrow, W.W.: Social networks and infectious disease: The colorado springs study. *Social science & medicine* 38(1), 79–88 (1994)
7. Neaigus, A., Jenness, S.M., Hagan, H., Murrill, C.S., Wendel, T.: Reciprocal sex partner concurrency and stds among heterosexuals at high-risk of hiv infection. *Journal of Urban Health*, 1–13 (2012)
8. Adogame, A.: Aids, religion, and the politics of social justice in sub-saharan africa. *The Wiley-Blackwell Companion to Religion and Social Justice*, 482–495 (2012)
9. Niazi, M.A.: Complex adaptive systems modeling: A multidisciplinary roadmap. *Complex Adaptive Systems Modeling* 1(1), 1 (2013)
10. Holland, J.H.: Studying complex adaptive systems. *Journal of Systems Science and Complexity* 19(1), 1–8 (2006)
11. Niazi, M.A., Hussain, A.: *Cognitive Agent-based Computing: Exploring Emergent Behavior in Complex Adaptive Systems using Agent-based Modeling and Complex Networks*. Springer (2012)
12. Niazi, M.A.: *Towards A Novel Unified Framework for Developing Formal, Network and Validated Agent-Based Simulation Models of Complex Adaptive Systems*. PhD thesis (2011)
13. Nations, U.: *Second generation surveillance for hiv: The next decade*. Technical report, HASP, Pakistan (2010)
14. Barré-Sinoussi, F., Chermann, J., Rey, F., Nugeyre, M., Chamaret, S., Gruest, J., Dauguet, C., Axler-Blin, C., Vézinet-Brun, F., Rouzioux, C., et al.: Isolation of a t-lymphotropic retrovirus from a patient at risk for acquired immune deficiency syndrome (aids). *Science* 220(4599), 868–871 (1983)
15. Bonabeau, E.: Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America* 99(suppl. 3), 7280 (2002)
16. Niazi, M., Hussain, A., Baig, A.R., Bhatti, S.: Simulation of the research process. In: *Simulation Conference, WSC 2008*, pp. 1326–1334. IEEE (Winter, 2008)

17. Siddiqua, A., Niazi, M., Mustafa, F., Bokhari, H., Hussain, A., Akram, N., Shaheen, S., Ahmed, F., Iqbal, S.: A new hybrid agent-based modeling & simulation decision support system for breast cancer data analysis. In: International Conference on Information and Communication Technologies, ICICT 2009, pp. 134–139. IEEE (2009)
18. Niazi, M.A., Siddique, Q., Hussain, A., Kolberg, M.: Verification & validation of an agent-based forest fire simulation model. In: Proceedings of the 2010 Spring Simulation Multiconference, Society for Computer Simulation International, p. 1 (2010)
19. Nagoski, E.: An agent based model of disease diffusion in the context of heterogeneous sexual motivation. PhD thesis, Indiana University (2006)
20. Atkinson, J.: A simulation model of the dynamics of hiv transmission in intravenous drug users. *Computers and Biomedical Research* 29(4), 338–349 (1996)
21. Wilensky, U.: Netlogo (1999)
22. Axelrod, R.: The complexity of cooperation: Agent-based models of competition and collaboration. Princeton Univ. Pr. (1997)
23. Fortino, G., Russo, W., Zimeo, E.: A statecharts-based software development process for mobile agents. *Information and Software Technology* 46(13), 907–921 (2004)
24. Cossentino, M., Fortino, G., Garro, A., Mascillaro, S.: Passim: a simulation-based process for the development of multi-agent systems. *International Journal of Agent-Oriented Software Engineering* 2(2), 132–170 (2008)

Analysis of Configurations for Photovoltaic Solar Energy Production Using Agent-Based Simulation

Jan Treur

VU University Amsterdam, Agent Systems Research Group
De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
j.treur@vu.nl
<http://www.cs.vu.nl/~treur>

Abstract. To configure a photovoltaic solar energy production plant the circumstances of a site play an important role. A site usually consists a number of specific locations that can be considered ideal (no shadow, perfect southward position, optimal vertical angle). However, much more often locations in a site are not ideal, to an extent that both depends on time of the year and time of the day. An important issue for decision making then is how much loss in efficiency a specific location will entail. In this paper this is analysed by an agent-based simulation method. Here photovoltaic modules with micro-inverters are conceptualised as autonomous energy producing agents, which are monitored by a central monitoring agent which also interacts with a user via a local and a global Web-based interface agent. The presented approach provides an analysis of the different locations at a site by simulating the agents over one full year with time steps of half an hour per day. The outcome of such a simulation provides an overview of the loss of efficiency for each of the locations depending on its characteristics with respect to shadow, orientation and vertical angle.

1 Introduction

To increase the fraction of renewable energy with energy production, the production of solar energy by photovoltaic panels is becoming more and more successful; e.g., [6, 7]. Many sites for solar energy plants are considered, both for domestic and business situations. In a first phase the focus is mostly on sites with only ideal locations (e.g., no shadow, perfect southward position, optimal vertical angle of the panels). Indeed, to configure a photovoltaic solar energy production plant, the circumstances of a site play an important role. However, to increase the available area for solar energy production, in practice more and more often sites are considered for which the locations are not all that ideal, depending on time of the year and time of the day. For example, in the summer months May, June and July not much shadow may occur, but in the other months shadow may occur to an extent that depends on the specific day in the year and hour of the day. How much loss in efficiency would this entail over a year? To design a plant configuration on a site, decisions have to be made on which locations the solar modules are placed. An important issue for this decision making process is how much loss in efficiency a specific location will entail,

compared to an ideal location. A trial and error approach would place the modules based on intuition or in an arbitrary manner, and after one year evaluate their results, after which reconfiguration of the site might take place; for example, see also [8]. In the current paper it is shown how this decision making process can be supported by an agent-based simulation method. This analysis can take place before actually building up the site, which has advantages over a trial and error approach.

The presented approach considers a multi-agent system based on photovoltaic modules together with their own micro-inverter, which are conceptualised as autonomous energy producing agents. They are monitored by a central monitoring agent which also interacts with a user via a local and a global Web-based interface agent. The presented approach provides an analysis of the different locations at a site by simulating the agents over one full year with time steps of half an hour per day. The outcome of such a simulation provides an overview of the loss of efficiency over the whole year for each of the locations depending on its characteristics with respect to shadow, orientation and vertical angle. This information supports the decisions on where to place the modules at the site.

In the paper, in Section 2 an overview is given of how a plant can be conceptualised and formalised as a multi-agent system, thereby using the concepts and formalisation of the agent system design method DESIRE; cf. [2, 3]. Section 3 discusses domain knowledge needed to make such agents realistic, and in Section 4 simulation results are discussed.

2 A Photovoltaic Installation as a Multi-agent System

In this section it is described how a given photovoltaic solar energy production site can be conceptualised and formalised as a multi-agent system; for a picture, see Fig. 1. As a source of inspiration an actual real life PV-system has been used, in an abstracted form. This system is based on Power One Aurora microinverters, a central monitoring unit Aurora CDD and local and global Web-based interfaces; see [9]. The PV site is assumed to be based on microinverters, which means that each solar panel has its own (micro) inverter which

- controls the panel's DC voltage to obtain an optimal level of generated power for the panel for given circumstances (e.g., irradiation and temperature): maximum power point tracking (MPPT); often hill climbing methods are used for this optimisation such as 'perturb and observe'.
- inverts the low voltage DC current (e.g., around 30V) into a high voltage AC current (e.g., 230V).

Together the panel and microinverter can operate in an autonomous manner, in parallel with (and independent of) the other panel-inverter pairs. These autonomous entities are conceptualised as Solar Production Agents (SPA). The goal of each of these SPA agents is to provide optimal power for the given combination of circumstances at each point in time. Note that the panel itself is not considered an agent as it is fully controlled by the microinverter.

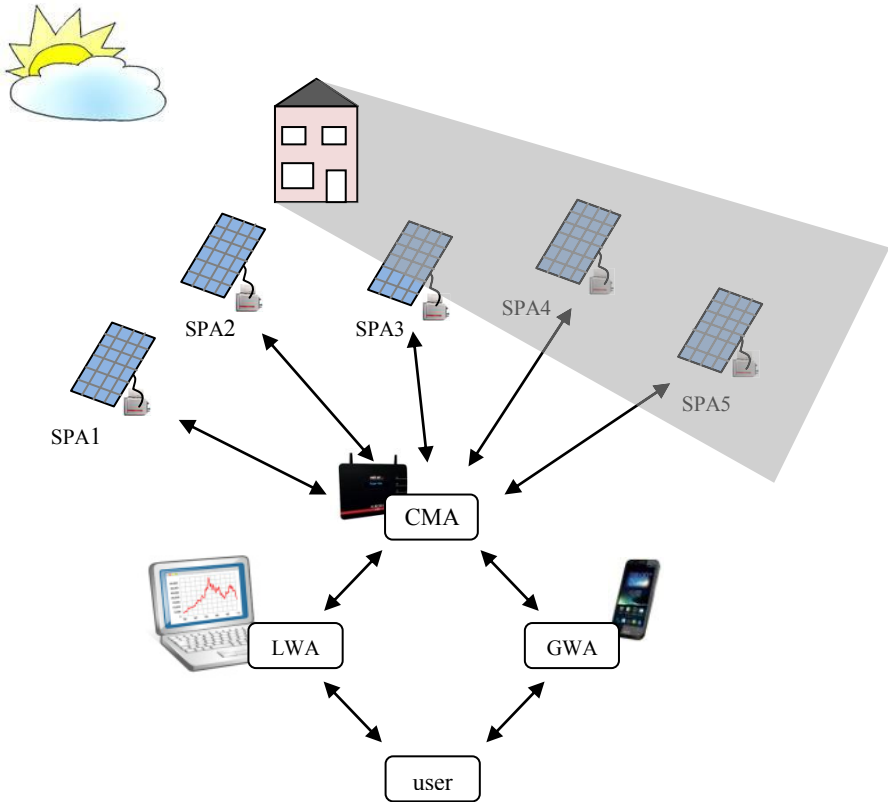


Fig. 1. Overview of the multi-agent system

The autonomy of the Solar Production Agents makes that they can easily adapt to the environmental circumstances, for example, of irradiation and temperature, in a manner independent of each other. This is in contrast with the also often used string-based approach, where a number of solar panels are combined in a serial manner into a string which gets only one inverter for the whole string. This makes the panels dependent in their responses on the environment; in particular, if only one panel of a string is in the shadow, then the whole string will have a very low production, also the panels that have no shadow. The approach based on micro-inverters considered here can adapt to environmental circumstances in a much more sensitive manner: a panel in the shadow will produce less, but this does not affect the production of the other panels which are not in the shadow.

To be able to get an overview of the whole plant, usually a Central Monitoring Agent (CMA) is used (in the example PV system the Aurora CDD unit). This is a device that communicates with each of the micro-inverters (wireless, by radio signals, for example, or by using specific wirings) and gets information from each of them. This information is obtained by the Solar Production Agents by observing their own energy production processes in the (physical) World. The Central Monitoring Agent

pro-actively gathers and maintains up to date information from of the SPA agents, by initiating a communication with them, for example each minute.

For an overview of the interactions within the multi-agent system, see Table 1. Note that the energy producing actions are initiated within the Solar Production Agents, but their execution takes place in the (physical) World component (after they were transferred to this component, from the initiating agent). This entails determining the effects (e.g., provided power P_{out}) of the action within World. In return observation information on these effects flows from World to the Solar Production Agents. Note that World is a system component here representing the physical world, but this component is not considered an agent. The agents interact with this component but this interaction concerns observation and action execution, not communication.

Table 1. Interaction structure for the multi-agent system

to from	World	SPA	CMA	LWA	GWA	user
World	-	Observation info	-	-	-	-
SPA	Action info	-	Observation info	-	-	-
CMA	-	Request info	-	Monitoring info	Monitoring info	Wifi or LAN info request
LWA	-	-	Request info	-	-	Observation info
GWA	-	-	Request info	-	-	Observation info
user	-	-	Wifi or LAN info	Request info	Request info	-

In addition the Central Monitoring Agent communicates with Local and Global Web-based interface Agents (LWA and GWA); this communication takes place locally through a local area or Wifi network and globally via Internet. These Web-based interface agents communicate with the user. The idea is that the Local Web-based interface Agent can be used within a local area network at home, for example at a PC or laptop, and that the Global Web-based interface Agent can be used via Internet anywhere, for example at a smartphone, as is the case for the Power One Aurora example considered.

The multi-agent system and its agents have been specified using the component-based agent design method DESIRE [1, 2, 3], and in particular the generic agent model GAM [4]. In each cell in Table 1 the name of an information type [3] is indicated. These information types include generic concepts for communication, observation and action performance from GAM; see [4]. In addition, for this specific instantiation of GAM they include domain-specific elements as shown in Table 2. Note that in the information type Action info the energy production action is specified. In the World the effect of this action has certain parameters, for example, the power P_{out} delivered. This

action effect will depend on the circumstances in the environment of the Solar Production Agent (e.g., the available irradiation) and the agent's physical characteristics such as the Watt peak $P_{panelpeak}$ and efficiency ρ_{panel} of its panel, the maximal input power $P_{invpeak}$ and the efficiency ρ_{inv} of the micro-inverter, and the angle and orientation of the panel (in Self info). This is specified also as part of World.

Table 2. Information types and their domain-specific information

<i>Information type</i>	<i>Domain-specific information included</i>	<i>Used in Agent</i>
Observation info	<ul style="list-style-type: none"> • P_{in} incoming power (W) • V_{in} incoming voltage (V) • processing temperature ($^{\circ}\text{C}$) • P_{out} outgoing power (W) • V_{out} outgoing voltage (V) • I_{out} outgoing current (A) • frequency (Hz) • alarm and warning states (e.g., ground leakage, communication faults) 	SPA World
Action info	<ul style="list-style-type: none"> • energy production action • shutting down (power off) • resetting (power off and restart) • production action effect (characteristics I_{out}, P_{out}, V_{out}) 	SPA World
Request info	Observation info with an extra label indicating that the information is requested	CMA, SPA, LWA, GWA
Monitoring info	Observation info with an extra label indicating a specific SPA agent name to which the info relates	CMA, LWA, GWA, user
Self info	<ul style="list-style-type: none"> • $P_{panelpeak}$ Watt peak of panel (W) • ρ_{panel} efficiency of the panel • $P_{invpeak}$ max power of inverter (W) • ρ_{inv} efficiency of the inverter • Vertical angle of panel ($^{\circ}$) • Horizontal orientation of panel ($^{\circ}$) • Serial numbers, MAC addresses • Software version information 	SPA
Agent info	Self info with an extra label indicating the agent to which the info relates	World, SPA, LWA, GWA, user
Collective info	Information on collective achievements: <ul style="list-style-type: none"> • P_{tot} total power produced by the plant (kW) • E_{tot} total energy produced over time by the plant (kWh) 	SMA, LWA, GWA, user

What has been presented up till now is an agent-external perspective on the multi-agent system, abstracting from what happens in agents internally. In addition to this external view, the component-based method DESIRE offers means to specify the agents from an internal perspective in a component-based manner. In particular the Generic Agent Model GAM [4] is composed of a number of standard components. In

this agent model the component *World Interaction Management* (WIM) takes care of interaction with the world, the component *Agent Interaction Management* (AIM) takes care of communication with other agents. Moreover, the component *Maintenance of World Information* (MWI) maintains information about the world, and the component *Maintenance of Agent Information* (MAI) maintains information about other agents. The processes involved in controlling the agent (e.g., determining, monitoring and evaluating its own goals and plans) but also the processes of maintaining a self model are the task of the component *Own Process Control* (OPC). In the component *Agent Specific Task* (AST) tasks specific for the agent can be modelled. For situations in which cooperation with other agents plays a role (such as in this case where the different Solar Production Agents together achieve the total production), in addition also a Cooperation Management (CM) component is included; cf. [1]. This model has been instantiated for each of the agents in the multi-agent system. As an example, in Table 3 it is shown which information types are used by the different components within the Central Monitoring Agent CMA. Note that in CMA the component World Interaction Management (WIM) is not used as this agent only communicates and has no own interaction with the world. In contrast, Table 4 shows the same for a Solar Production Agent SPA.

Table 3. Overview of the components within the agent CMA and the information types used

Internal agent concepts	CMA	Component
<i>World Model</i>	Monitoring information	MWI
<i>Agent Models</i>	Monitoring info Agent identification info	MAI
<i>Collective Model</i>	Collective info	CM
<i>Communication initiation</i>	Monitoring info Request info	AIM
<i>Processing of received communication</i>	Monitoring info	AIM

Table 4. Overview of the components within an agent SPA and the information types used

Internal agent concepts	SPA	Component
<i>World Model</i>	Monitoring information	MWI
<i>Self Model</i>	Self info	OPC
<i>Action initiation</i>	Action info	WIM
<i>Communication initiation</i>	Monitoring info	AIM
<i>Processing of received observation results</i>	Monitoring info	WIM
<i>Processing of received communication</i>	Request info	AIM

Here the component World Interaction Management is used for observation and for performing the action energy production, but not the components Cooperation Management (CM) and Maintenance of Agent Information (MAI).

3 Situational Efficiency of the Solar Production Agents

To be able to perform agent-based simulation experiments, knowledge has been modelled about how at any point in time, the efficiency of the energy production action of a Solar Production Agent depends on circumstances. In general, the action effect of the energy production action of a Solar Production Agent (as determined in the World component) on provided power P_{out} can be described as a function of:

- used irradiation, which itself depends on
 - the efficiency ρ_a due to angle and orientation of the panel
 - the efficiency $\rho_s(t)$ due to shadow at time t
 - the available irradiation $irr(t)$ at time t
- the maximal power $P_{panelpeak}$ of the panel (Watt peak)
- the efficiency ρ_{panel} of the panel
- the maximal power $P_{inverterpeak}$ of the micro-inverter
- the efficiency ρ_{inv} of the micro-inverter.

The following relations are assumed:

- Provided power P_{panel} by panel: $\min(P_{panelpeak}, \rho_a \rho_s(t) irr(t)) \rho_{panel}$
- Provided power P_{out} by inverter: $\min(P_{inverterpeak}, P_{panel}) \rho_{inv}$

In the model the World component receives the energy production actions from the Solar Production Agents, and determines the action effects P_{out} for each of these Solar Production Agents based on the above formula. In turn the Solar Production Agents receive (as observation) this effect of their own action (e.g., P_{out}) from the World component and communicate this to the Central Monitoring Agent CMA. Note that ρ_{panel} , ρ_{inv} , $P_{panelpeak}$, and $P_{inverterpeak}$ are given characteristics of the Solar Production Agent, represented as Self info in each Solar Production Agent and as Agent info for all Solar Production Agents in the World component. The other three $\rho_s(t)$, ρ_a , and $irr(t)$ are variables that can be manipulated or depend on time of the day and year; they will be discussed in more detail in this section.

3.1 Modelling Shadow Effects on the Energy Production: The Variable $\rho_s(t)$

As a first step it has been modelled how shadow affects the results of the energy production action (the variable $\rho_s(t)$). As shadows directly relate to obstacles on the one hand, and positions of the sun on the other hand, they vary much with the sun positions at different times of the day and at different days of the year. The position of the sun (seen from the earth) is characterised by two angles (see also Christensen and Barker, 2001):

- The vertical angle above the horizon
- The horizontal angle with the direction of North

The *vertical sun angle* dynamics as seen from the earth has been modelled (as an approximation) by

$$vsa(t, d) = 23.4 \cos(360(d-172)/365.26) + (90-nl)\cos(360(t-t0)/24)$$

when this is positive and $vsa(t, d) = 0$ otherwise

Here

$vsa(t, d)$ = vertical sun angle at time t of day d

t = time on the day

d = day of the year

nl = northern latitude (= 52.7 in the simulation)

el = easter longitude (= 4.7 in the simulation)

$t0 = \text{round}(2*(13-el/15);0)/2 = 12.5$ (= time of maximal vertical sun angle)

The first term in this formula varies from -23.4 (on December 21, day 355) to 23.4 (on June 21, day 172) over the year. The second term is for $t = t0$ always $90-nb$. Therefore for $nb = 52.7$ the maximal vertical sun angle varies from 14 (on Dec. 21, day 355) to 61 (on June 21, day 172), which indeed is empirically valid for the given site. For $t = t0-6$ and $t = t0+6$ the second term is 0. The *horizontal sun angle* depends in a linear manner on the time t of the day:

$$has(t) = \text{horizontal sun angle (with North)} = 180 + (t-t0)*15$$

Given this model for the sun's dynamics, as a next step it has been modelled how for a given obstacle (assumed here to be a rectangle with a certain position, height and direction) this results in a certain *shadow length* $sl(t, d)$:

$$\begin{aligned} sl(t, d) &= oh \sin(sao(t, d))/\tan(vsa(t, d)) && \text{if } vsa(t, d) > 0 \text{ and} \\ & && \sin(sao(t))/\tan(vsa(t, d)) > 0 \\ &= 0 && \text{if } vsa(t, d) > 0 \text{ and} \\ & && \sin(sao(t))/\tan(vsa(t, d)) \leq 0 \\ &= maxs && \text{if } vsa(t, d) = 0 \end{aligned}$$

Here

$sl(t, d)$ = shadow length at time t of day d

oh = obstacle height

$sao(t)$ = sun angle with obstacle

$sao(t) = has(t) - 90 - ao$

ao = angle obstacle with East-West direction (= 25 in the simulation)

$maxs$ = max length shadow

Given this, the irradiation loss due to shadow has been modelled. Note that the potential irradiation/hour is given by

$$\begin{aligned} pr(t) &= \text{potential radiation /hour} = \sin(vsa(t)) && \text{if } vsa(t) > 0 \\ &= 0 && \text{otherwise} \end{aligned}$$

The irradiation loss due to shadow can be modelled in different manners, for example, by identifying shadow areas of obstacles using, their dimensions and distances, or by determining the vertical angles of the contours of the horizon in all directions. For the simulations it has been modelled in the first manner:

$$rl(t, d) = \begin{cases} \sin(vsa(t, d)) (sl(t, d)-cd)/(fd-cd) & \text{if } cd < sl(t, d) < fd \\ \sin(vsa(t, d)) & \text{if } sl(t, d) \geq fd \\ 0 & \text{otherwise} \end{cases}$$

Here

- $rl(t, d)$ = irradiation loss at t and d
- cd = closest distance of location from obstacle
- fd = farrest distance of location from obstacle

The distribution over a day has been aggregated to a day loss and month loss as follows:

$$\begin{aligned} drl(d) &= \text{day irradiation loss} = \int_0^{24} rl(t, d) dt \\ dpr(d) &= \text{day potential irradiation} = \int_0^{24} pr(t, d) dt \\ mrl(m) &= \text{month irradiation loss} = \int_0^{30 \times 24} rl(t, m) dt \\ mpr(m) &= \text{month potential irradiation} = \int_0^{30 \times 24} pr(t, m) dt \\ dlf(d) &= \text{day irradiation loss / day potential irradiation} = drl(d) / dpr(d) \\ mlf(m) &= \text{month loss fraction} = mrl(m) / mpr(m) \end{aligned}$$

These models provide a way to keep track in simulations of the overall effects in loss of efficiency due to shadow.

3.2 Modelling the Effects of Panel Orientation and Angle on the Energy Production: The Variable ρ_a

A next circumstance addressed is the orientation and angle under which the panel of a Solar Production Agent is positioned (the variable ρ_a). For example, at [10] a schematic overview is shown of how efficiency relates to the vertical and horizontal angle of a panel, taken over a year. This has been modelled here as an approximation

$$\rho_a(\alpha) = \gamma \sin(90 + \alpha - \alpha_{opt}) + \eta$$

with α the vertical angle and α_{opt} the optimal vertical angle. When 35° is the optimum, as approximately is the case in middle European areas, it follows that

$$\rho_a(\alpha) = \gamma \sin(\alpha + 55) + \eta$$

The other parameters can be determined depending on the horizontal orientation, for example:

- Orientation to south: $\rho_{a,s}(\alpha) = 0.65 \sin(\alpha + 55) + 0.35$
- Orientation to south south west: $\rho_{a,ssw}(\alpha) = 0.58 \sin(\alpha + 55) + 0.39$

The latter was used in the example plant simulation, which has an orientation south south west.

3.3 Modelling the Effects of Available Irradiation on the Energy Production: the Variable $irr(t)$

For input on irradiation for different times in a year, realistic empirical data were acquired (from the Dutch Meteorological Institute KNMI). These data show the distribution of irradiation over different months of the year, indicated as $irr(m)$ with m a month. Note that such information for Europe can also be obtained at [11]. From these irradiation figures the relative distribution of *irradiation fractions* has been made (total sum = 1), indicated by $irrf(m)$ with m a month:

$$irrf(m) = irr(m) / \sum_{m'=1}^{12} irr(m')$$

To get an impression of the different periods of the year, the diagram shown in Fig. 2 has been made. As can be seen there, the months May, June and July provide 45% of the year (red numbers) irradiation. From March to September 85% is provided. This has a strong relation with the maximal vertical sun peak angles (black numbers). This overview is of some help in interpreting the results from the simulations.

month	fraction	sun peak	
Jan	1 0.018	0.98	18°
Feb	2 0.039	0.94	27°
March	3 0.069	0.85	38°
April	4 0.110	0.69	50°
May	5 0.144	0.45	58°
June	6 0.161	58°	61°
July	7 0.143	50°	58°
Aug	8 0.132	38°	50°
Sept	9 0.091	27°	38°
Oct	10 0.053	18°	27°
Nov	11 0.024		18°
Dec	12 0.015		15°

Fig. 2. Relative contribution of irradiation of different time periods in the year

4 Simulation Experiments

The agent-based models described in Sections 2 and 3 enable to simulate a designed plant configuration, for example, for one year, before the plant is to be realised. A period of one, or even multiple years is needed as the circumstances are different at different times of the year, and even between different years. Moreover, within each day simulation took place with time steps of half an hour, in order to deal with

different circumstances during the day. Given the obtained empirical data on irradiation per month (see Fig. 2), the decision was made to run simulations by taking one day for each month. More specifically, as an approximation, 12 days d_1, \dots, d_{12} in a year were simulated, representing the 12 months (taken at the 21-th of each month); the month loss fraction for month m is based on the day loss fraction (see Section 3) of the day d_m representing this month:

$$mlf(m) = dlf(d_m)$$

From this the year loss fraction was determined, using the irradiation fraction:

$$\text{year loss fraction} = \sum_{m=1}^{12} mlf(m) irrf(m) = \sum_{m=1}^{12} dlf(d_m) irrf(m)$$

Simulation results for an example site are shown in subsequent Figures 3 and further. Note that the figures shown focus on loss fractions, as these are most relevant for the decision making. The example site consists of four locations with different shadow circumstances, with the following characteristics.

- Location 1 shelter in the garden with an almost flat roof oriented south south west with shadow from the house with height 5.00 meter at a distance of 6.00 to 10.00 meter south east
- Location 2 garage with a flat roof oriented south south west with shadow from the house with height 5.00 meter at a distance of 2.85 to 4.50 meter south east
- Location 3 flat roof at top of a dormer oriented south south west with shadow from a row of trees east south east at a distance of 8.50 to 15.50 meter and height 5.50 meter and from the rooftop south south west at a distance of 0.85 to 3.00 meter with height 0.50 meter
- Location 4 sloped roof oriented south south west with shadows from a row of trees east south east at a distance of 8.50 to 15.50 meter and height 5.50 meter

First the panels are assumed under the ideal angles. In Fig. 3 the month loss fractions $mlf(m)$ are shown for the different locations.

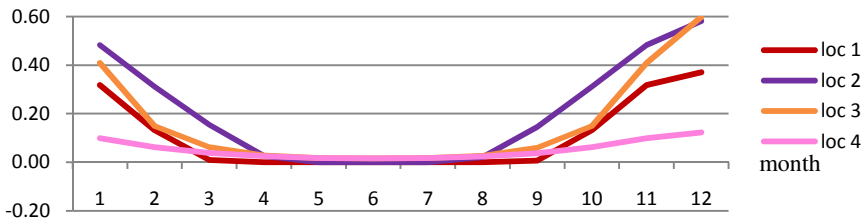


Fig. 3. Month loss fractions over months for the different Solar Production Agents

It can be seen that in the summer months the loss fraction of all locations is practically 0 (almost no shadows). But in other times of the year (from September to April) there are differences up to 500%. In Fig. 4 the losses are related to the year instead of the month by multiplying the month loss fraction with the month irradiation fraction: $mlf(m) irrf(m)$. Also here it can be seen that in the months from September to April large differences occur.

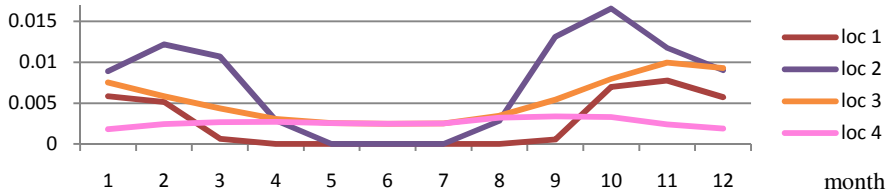


Fig. 4. Year loss fractions over months for the different Solar Production Agents

In Fig. 5 the total irradiation used for the different locations is shown, still assuming an ideal angle of the panels. It can be seen that in the summer months the used irradiation is practically the same for all locations (almost no shadows). But in other times of the year there are differences. However, in these times the overall available irradiation is less, so although the loss is a high fraction, in absolute terms the differences are more modest.

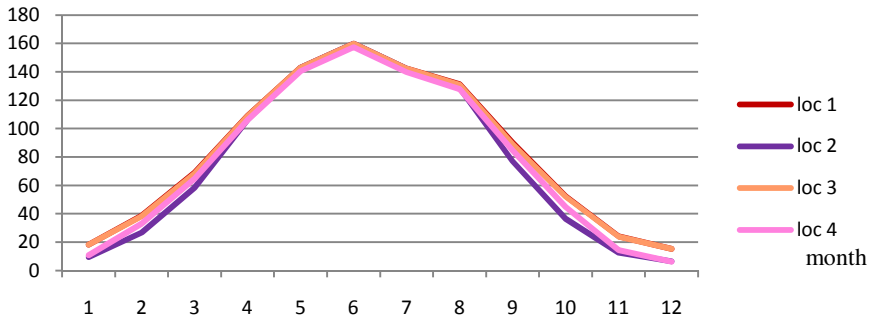


Fig. 5. Irradiation used at each location in each month

In Fig. 6 the overall loss fractions for the year as a whole are shown. The four locations differ in losses from 3% to almost 9%. Such differences are worthwhile to consider in deciding where to place the Solar Production Agents.

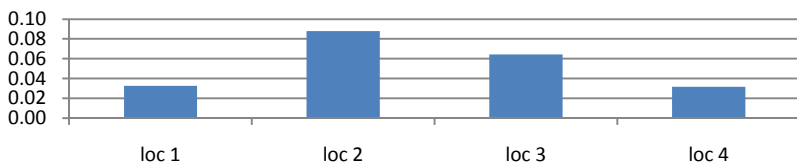


Fig. 6. Year loss fractions for the different locations for equal, ideal angle of the panels

In Fig. 7 results are shown for a different simulation in which for each location there is a different vertical angle of the panel. Due to the less optimal angle year

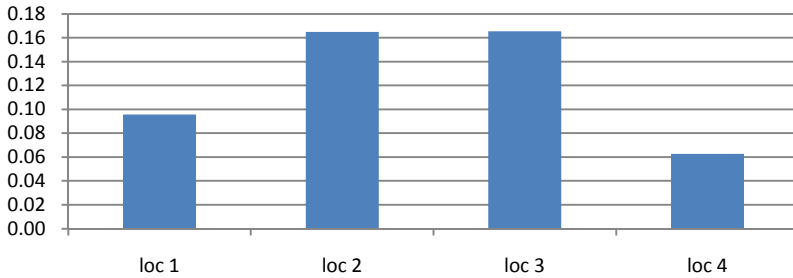


Fig. 7. Year loss fractions for the different locations for different angles: loc 1: 15° loc 2: 10° loc 3: 5° loc 4: 30°

losses become more in this case, up to 16%, and the distribution changes as well. For example, location 3 becomes much worse due to a less ideal angle of 5° .

5 Discussion

Usually the performance of photovoltaic solar energy production plants (e.g., [6, 7]) are analysed during their operation (e.g., [8]). A plant is distributed over a number of locations within the site. Such locations in a site are not always ideal, and may entail loss of efficiency due to effects of shadow and/or to nonoptimal orientations and angles for the panels (which may, for example, depend on slopes and orientation of available roofs). It is not easy to estimate the effects of such circumstances at forehand, as they vary with time of the day and day of the year. Therefore, evaluation of performance of an operational plant makes most sense after at least one year, but even then the weather circumstances in that year may not be representative for other years. From such evaluations afterwards it may be found that the configuration of the plant can better be changed by moving panels from apparently less favourable locations to more favourable locations, but in the meantime more than a year was lost.

In this paper a different approach was followed: the expected performance of possible configurations of a plant are analysed at forehand by an agent-based simulation method. Using the outcomes of such a what-if analysis, there is a better chance that the configuration of a plant uses the best locations right from the start. The photovoltaic modules with micro-inverters were conceptualised as autonomous energy producing agents, interacting with a central monitoring agent, which in turn interacts with a user via a local and a global Web-based interface agent. The presented approach has been shown to provide an analysis of the different locations at a site by simulating the agents over one full year with time steps of half an hour per day, which easily can be refined, for example, to smaller time steps of, for example, 5 minutes. Based on this analysis a decision can be made about the most optimal locations for the modules.

The aim of the work presented here was to bring together knowledge from two different disciplines. On the one hand this concerns knowledge about modelling

parallel processes in a conceptual and formal agent-based framework. On the other hand detailed domain knowledge was considered about photovoltaic solar energy production and all kinds of practical factors affecting the efficiency of it. Such domain knowledge is often hidden in tools used in practice. For example, see [12] for an overview of such tools, and [13] for one specific tool: Solar Pathfinder. Although the details are not revealed, it seems that in the latter tool the role of shadow is determined based on the contours of the horizon. As also mentioned in Section 3.1 above, the method used in the work presented here takes a different approach, not based on contours, but on the dimensions and distances of the obstacles. In the paper it is shown how the agents considered as conceptual entities to model the parallel processes can be provided with such very detailed domain knowledge in order to obtain a level of detail needed for realistic simulations of real world situations.

References

1. Brazier, F.M.T., Cornelissen, F., Jonker, C.M., Treur, J.: Compositional Specification of a Reusable Co-operative Agent Model. *International Journal of Cooperative Information Systems* 9, 171–207 (2000)
2. Brazier, F.M.T., Dunin Keplicz, B., Jennings, N., Treur, J.: DESIRE: Modelling Multi-Agent Systems in a Compositional Formal Framework. *International Journal of Cooperative Information Systems* 6, 67–94 (1997)
3. Brazier, F.M.T., Jonker, C.M., Treur, J.: Principles of Component-Based Design of Intelligent Agents. *Data and Knowledge Engineering* 41, 1–28 (2002)
4. Brazier, F.M.T., Jonker, C.M., Treur, J.: Compositional Design and Reuse of a Generic Agent Model. *Applied Artificial Intelligence Journal* 14, 491–538 (2000)
5. Christensen, C.B., Barker, G.M.: Effects of Tilt and Azimuth on Annual Incident Solar Radiation for United States Locations. In: *Proc. of Solar Forum 2001 - Solar Energy: The Power to Choose*, Washington, DC, April 21–25 (2001)
6. Eltawil, M.A., Zhao, Z.: Grid-connected photovoltaic power systems: Technical and potential problems—A review. *Renewable and Sustainable Energy Reviews* 14, 112–129 (2010)
7. Parida, B., Iniyar, S., Goic, R.: A review of solar photovoltaic technologies. *Renewable and Sustainable Energy Reviews* 15, 1625–1636 (2011)
8. Sharma, V., Chandel, S.S.: Performance analysis of a 190 kWp grid interactive solar photovoltaic power plant in India. *Energy*, 1–10 (2013), doi:10.1016/j.energy.2013.03.075
9. URL Power One Aurora, <http://www.power-one.com/renewable-energy/products/solar/string-inverters/panel-products/aurora-micro/series-0>
10. URL The effect of vertical angle and horizontal orientation over the year, http://www.enpower.be/finance/home/enpower_en.php?lnk=zon&page=techniek_zon
11. URL Irradiation data, <http://re.jrc.ec.europa.eu/pvgis/apps4/pvest.php>
12. URL Overview of available tools, <http://www.builditsolar.com/References/SunChartRS.htm>
13. URL Solar Pathfinder, <http://www.solarpathfinder.com>

Dynamic Allocation of a Domestic Heating Task to Gas-Based and Heatpump-Based Heating Agents

Jan Treur

VU University Amsterdam, Agent Systems Research Group
De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
j.treur@vu.nl
<http://www.cs.vu.nl/~treur>

Abstract. In this paper a multi-agent model for a domestic heating task is introduced and analysed. The model includes two alternative heating agents (for gas-based heating and for heatpump-based heating), and a third allocation agent which determines the most economic allocation of the heating task to these heating agents over the days in a year. For allocation decisions it is analysed how the performance of a heatpump depends on the outdoor temperature. One method discussed is a what-if analysis method using agent-based simulation, another method is by mathematical analysis to derive more precise knowledge about the most optimal allocation choice. These methods can be used by the allocation agent to determine in a dynamic, adaptive manner per day a most economic allocation, depending on the (predicted) outdoor temperature.

1 Introduction

Especially in more northern countries, a substantial amount of domestic energy use during the winter season concerns heating. Often the heating systems used are based on not renewable resources such as gas and oil, which over the years are rapidly becoming more expensive. Moreover, as a byproduct, serious negative effects on the climate are obtained. For these reasons more and more often, alternative domestic heating systems are considered. An often considered alternative is the use of a heatpump (e.g., [1, 7, 9, 12, 15], which takes thermal energy from the environment (from air, water or soil) and uses this to heat the water of a central heating system in the house. However, there are some drawbacks.

One issue is that water is often not available in the direct neighbourhood, so then this source is excluded. Another issue is that to use thermal energy from the soil often a serious financial investment is needed, whereas a heatpump by itself is usually already a much more expensive investment than a gas- or oil-based heating system. Therefore often a heatpump is considered which takes thermal energy from the air (air to water heatpump). Another main issue here is that at the coldest days, when heating needs most energy, the air temperature is low, and due to that an air to water heatpump becomes less economic in use, or even lacks capacity to achieve the heating of the water of the heating system to the required level.

Due to these drawbacks in many cases a more feasible option considered is to combine different heating systems and to allocate the heating task in a dynamic manner to one of these systems, depending on the circumstances, in particular the outdoor temperature. For example, a gas- or oil-based system can be allocated the heating task only on the coldest days (their efficiency does not depend on the outdoor temperature), whereas for all other days a heatpump is allocated. This already can save money and reduce the negative effects on the climate. It is such a hybrid configuration that is considered here.

In a hybrid system it is crucial to make the right allocation at the right moment in a dynamic, adaptive manner. In this paper this is modelled and analysed from an agent-based perspective. Two alternative heating systems are considered as heating agents, and a third allocation agent determines the most economic allocation of the heating task to these agents over the days in a year; this setup will be introduced in Section 2. To be able to make good allocation decisions a number of aspects have to be modelled. First, it has to be known how the performance of a heatpump depends on the outdoor temperature; this is addressed in Section 3. In Section 4 a model is introduced and simulations with this model over a year period are analysed, based on realistic data from 2012. In Section 5 by a mathematical analysis more precise knowledge is obtained about the most optimal allocation choice.

2 General Setup

The modelling setup addressed in this paper follows what is often done in hybrid heating systems in practice, and has the form of an agent-based model based on three agents: the Gas-based Heating Agent (GHA), the Heatpump-based Heating Agent (HHA), and the Heating Allocation Agent (HAA); see Fig. 1. Each of the first two agents can take care of the heating via control of the water temperature of the central heating system. They are responsive for input communicated by the Heating Allocation Agent.

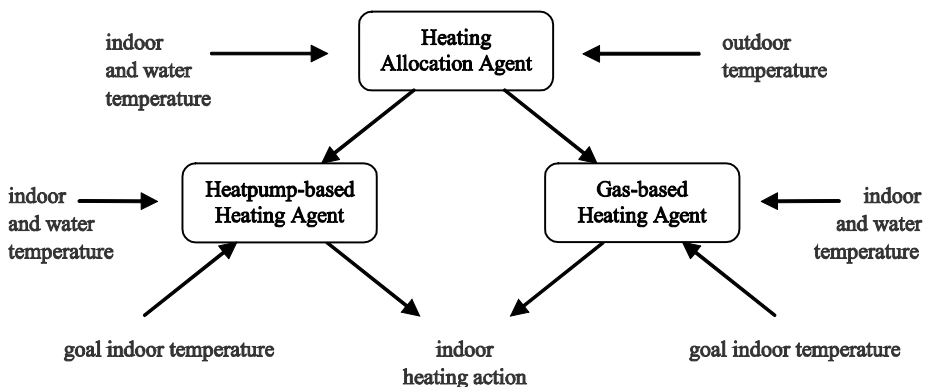


Fig. 1. Overview of the setup with the three agents

The allocation agent has as its goal to generate smart decisions (on the fly) about to which of the other agents to allocate the task of heating, and to communicate these decisions to both agents. Its main source of information is acquired by sensing and monitoring the outdoor temperature and the indoor air and water temperature. It uses knowledge about how economic the two heating agents are for different outdoor temperatures. Especially the efficiency (performance factor) of heating by a heatpump strongly depends on the outdoor temperature. For each given (predicted) outdoor temperature the allocation agent allocates the most economic heating agent to the heating task. So, more specifically its goal is to obtain a most economic performance of the heating task,

Each of the heating agents HHA and GHA receives the allocation information communicated by the allocation agent. Each of them responds to that communication by actually performing the heating when the communicated information indicates that it is allocated, and by being idle when this information indicates it is not allocated. As the heating agents are purely responsive, most of the intelligence concerning optimisation of the heating by this 3-agent system lies in the allocation agent HAA. To achieve its goal it needs detailed domain knowledge. In subsequent sections the required domain knowledge is discussed and it is analysed how the overall system works based on this knowledge.

The design of the three agents is based on the component-based Generic Agent Model (GAM) presented in [6], designed according to the component-based agent system design method DESIRE [4, 5]. Within this model GAM the component *World Interaction Management* takes care of interaction with the world, the component *Agent Interaction Management* of communication with other agents. Moreover, the component *Maintenance of World Information* maintains information about the world, and the component *Maintenance of Agent Information* information about other agents. The processes involved in controlling the agent and of maintaining a self model are the task of the component *Own Process Control*. In the component *Agent Specific Task*, specific tasks of the agents can be modelled.

For the two heating agents GHA and HHA the component Agent Interaction Management handles the communication with the Heating Allocation Agent AA and with the human(s) in the house (via the thermostat as a communication mean). The received allocation information is stored as self information in Own Process Control and as the heating agents are purely responsive, from there immediately this information flows to World Interaction Management (if the allocation information expresses that the agent has been allocated the task), resulting in generation of the heating action. The heating action itself also depends on information the heating agent perceives about the indoor temperature and the water temperature. This is received in World Interaction Management and as an intermediate step stored in Maintenance of World Information. Moreover, the information about the goal indoor temperature communicated by the humans in the house is taken into account. This communication

is handled via Agent Interaction Management; as an intermediate step the received goal information is stored in Maintenance of Agent Information. The Heating Allocation Agent HAA involves more complex processing in its Agent Specific Task (which is heating allocation). This task will be addressed in more detail in subsequent sections. The other components function in a way similar to how they function in the heating agents. World Interaction Management and Maintenance of World Information take care of receiving and storing world information about indoor and outdoor temperature and water temperature of the heating system. Agent Interaction Management and Maintenance of Agent Information take care of communication with the heating agents and storing the allocation information involved.

For the Agent Specific Task heating allocation the agent HAA needs to perform an analysis of the expected costs of the two heating agents. First of all, in order to be able to compare the two heating systems on efficiency it needs a way of estimating the seasonal performance factor of the heatpump-based heating system for given circumstances. This is addressed in Section 3. Next it needs methods to assess in a comparative manner how economic the two heating systems are. This can be done in (at least) two different ways. The first method, discussed in Section 4 is by an agent-based what-if analysis (simulation). The Heating Allocation Agent could incorporate such a what-if analysis in its Agent Specific Task component in order to make allocation decisions. This provides a more elaborate variant of this agent. The second method, discussed in Section 5 is by mathematical analysis. Results of such a mathematical analysis can be used as a form of compiled knowledge in the Agent Specific Task component of the Heating Allocation Agent. This provides a more concise variant of this agent.

3 Estimation of Seasonal Performance Factors for a Heatpump

The seasonal performance factor SPF strongly depends on the water temperature of the heating system and the outdoor temperature. Manufacturers often give indications of these performance factors for just a few water and outdoor temperatures. However, to determine the electricity use of a heatpump over a year, it is needed to have an estimation of SPF for the given water temperature and each possible outdoor temperature, as this outdoor temperature shows much variation over the year. To obtain a reasonable estimation of how for a given water temperature the performance factor depends on the outdoor temperature, theoretical analyses can be made. However, such theoretical analyses are often not guaranteed to provide values that occur in reality. A different route is to take empirical data as a point of departure and make an approximation of them by a mathematical function. A useful source of such data can be found at [16]. In Fig. 2 a graph is shown with values from this Website for the average day temperature on the horizontal axis and the performance factor on the vertical axis (for water temperature approximately 50°C).

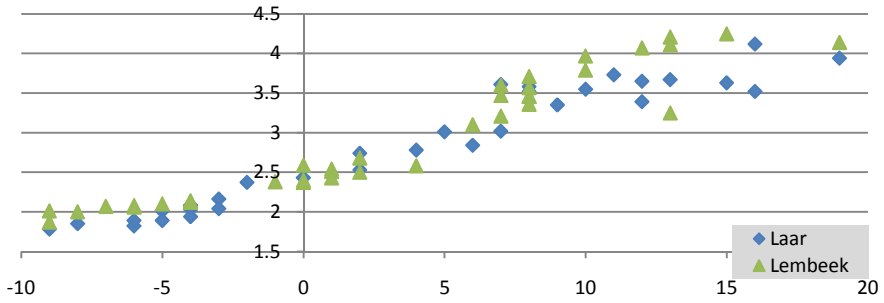


Fig. 2. Empirical data on seasonal performance factors in relation to outdoor temperature over 2012 for sites in Lembeek and Laar (water temperature 50°C)

More specifically, this has been done for the sites at Lembeek and Laar, where the General Waterstage HT heatpump combination WH16/WOH16 is used. In Fig. 3 a linear approximation of SPF for the interval from -10°C to +20°C is drawn; this is assumed of the form

$$SPF(T_{od}) = 7.5 - 0.1 * (T_{water} - T_{od}) \quad \text{with } T_{water} = 50$$

This linear approximation suggests a *rule of thumb* stating:

Every degree Celsius lower in outdoor temperature makes the performance factor SPF drop by 0.1.

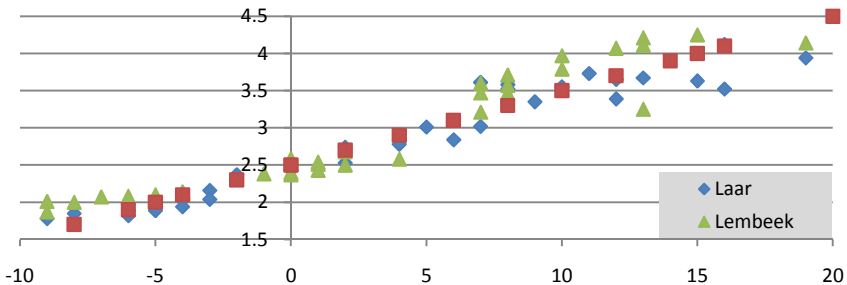


Fig. 3. Linear approximation of seasonal performance factors in relation to outdoor temperature compared to empirical data over 2012 for sites in Lembeek and Laar (water temperature 50°C)

Using this approximation, the seasonal performance factor can be estimated throughout a year, when the day temperatures are given. For example, in Fig. 4 in the upper graph the (average) day temperatures (in De Bilt, The Netherlands) of all days of 2012 are given, and in the lower graph the seasonal performance factor is estimated for all these days based on the linear approximation.

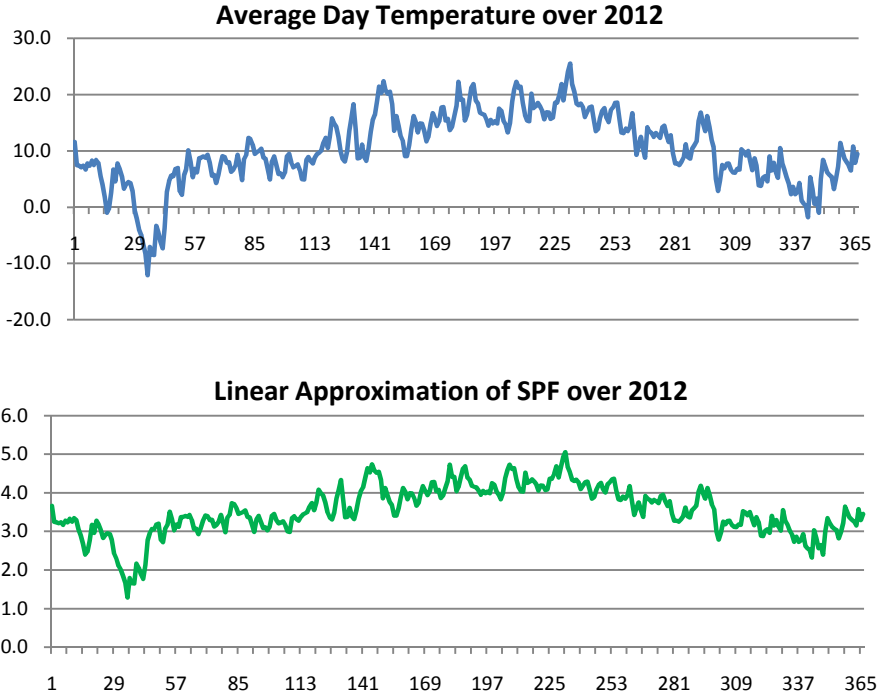


Fig. 4. Empirical data on average day temperature in De Bilt, the Netherlands (upper graph) and estimation of seasonal performance factors over 2012 based on the linear approximation (lower graph)

From Fig. 4 it can be seen that for outdoor temperatures from 11°C and higher, the values of SPF have much more variation than for the lower outdoor temperatures. Therefore any approximation, including the linear approximation introduced above, may show relatively high deviations in the interval above 10°C. If the values for outdoor temperatures from 11°C and higher are neglected, a more close inspection of the remaining interval from -10°C to +10°C reveals a pattern of a slightly bended upward curve with empirical values closer to -10°C to +10°C that are higher than the linear approximation (which fits best in the middle area of this interval, say from -5°C to +5°C). This curve can be described in the interval from -10°C to +10°C by a quadratic pattern as a more accurate approximation than the linear one, as is shown in Fig. 5. The quadratic pattern shown is described by

$$SPF(T_{od}) = 7.45 - 0.1 * (T_{water} - T_{od}) + 0.004 T_{od}^2 \quad \text{with } T_{water} = 50$$

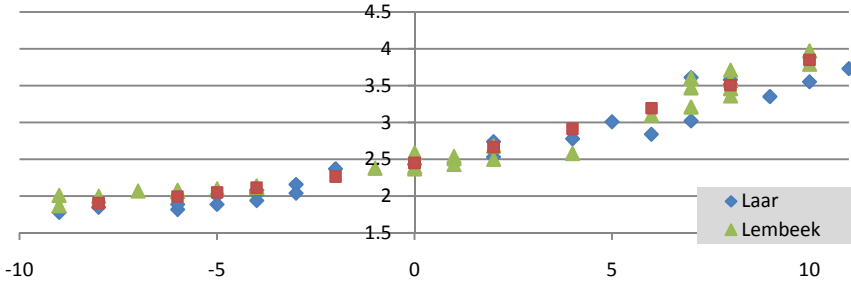


Fig. 5. Quadratic approximation of seasonal performance factors in relation to outdoor temperature compared to empirical data over 2012 for sites in Lembeek and Laar (water temperature 50°C)

4 What-If Analysis Using an Agent-Based Simulation Model

As a first method for the heating allocation agent to analyse the costs of energy use for heating for the two heating agents, what-if analysis by agent-based simulation is used. The simulations make use of the three agents introduced in Section 2: the gas-based heating agent GHA, the heatpump-based heating agent HHA, and the heating allocation agent HAA. Fig. 6 shows the variables used in the model used and the dependencies between them, and Table 1 summarizes them. The heating to be performed by the agents is responsive for the circumstances in the environment, in particular, for the average outdoor day temperature T_{od} . This is assumed given (either obtained by sensing, or as a prediction from a weather forecast). A general format to determine how much energy is to be provided to the heating system makes use of the concept of *degree day*, denoted by dd . This concept is based on the assumption that the amount of energy needed to maintain a difference in temperature (between indoor and outdoor) is proportional to this difference (e.g., [13]). The number of degree days for a given day t directly relates to the difference between the outdoor and the (average) indoor day temperature T_{id} (when the latter is higher than the former), and else is 0:

$$dd(t) = \begin{cases} \sigma(t) * (T_{id}(t) - T_{od}(t)) & \text{when } T_{id}(t) > T_{od}(t) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Here $\sigma(t)$ is as seasonal correction weight factor which is 1.1 for the months November, December, January and February, 1 for the months March and October, and 0.8 for the months April, May, June, July, August and September.

For a period consisting of a number of days these degree days are simply added. For a gas-based heating system the total cost of heating, for example, during a year is proportional to the number of degree days in the following manner. First, for each

degree day an amount η of gas (in m^3) is needed. Therefore the gas-based provision $gp(t)$ (m^3 gas used for day t) is determined as (see also Fig. 6, left hand side):

$$gp(t) = \eta dd(t)$$

Next, assuming that every m^3 gas costs π_{gas} euro, the costs $gc(t)$ of gas for day t is given by

$$gc(t) = \pi_{gas} gp(t) = \pi_{gas} \eta dd(t) \quad (2)$$

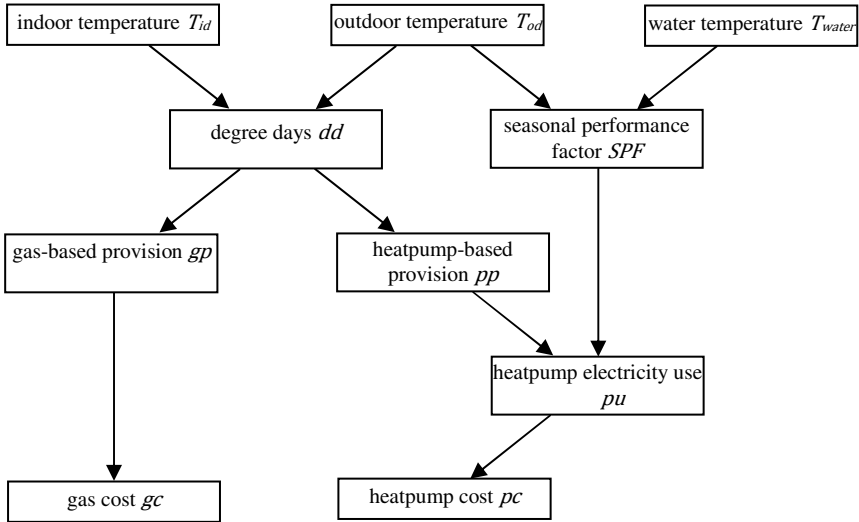


Fig. 6. Dependencies of the variables in the model

For the costs of heating based on a heatpump a similar but slightly more complex model is used. First, for each degree day an amount ε of electricity (in kWh) has to be provided. Therefore the heatpump-based provision $pp(t)$ (kWh provided for heating at day t) is determined as (see also Fig. 6, right hand side):

$$pp(t) = \varepsilon dd(t)$$

Note that this is the amount provided but not the amount $pu(t)$ used by the heatpump itself, as part of the provided energy $pp(t)$ comes from the environment. This is expressed using the seasonal performance factor SPF as follows:

$$pu(t) = pp(t) / SPF(T_{od}(t)) = \varepsilon dd(t) / SPF(T_{od}(t))$$

Finally, assuming that one kWh electricity costs π_{el} euro, the costs $pc(t)$ of heating for day t by the heatpump is given by

$$pc(t) = pu(t) * \pi_{el} = (\pi_{el} \varepsilon / SPF(T_{od}(t))) dd(t) \quad (3)$$

Table 1. Main concepts

notation	description		unit	values used
SPF	seasonal performance factor			
$T_{od}(t)$	average outdoor temperature at day t		$^{\circ}C$	
$T_{id}(t)$	average indoor temperature at day t		$^{\circ}C$	
$T_{water}(t)$	average water temperature of the heating system at day t		$^{\circ}C$	
$\sigma(t)$	seasonal weight factor	0.8 for months 4-9, 1 for months 3 and 10 , 1.1 for months 11-2		
$dd(t)$	degree days at day t		dd	
$gp(t)$	gas provision for day t		m^3	
$pp(t)$	heat pump energy provision for day t		kWh	
$pu(t)$	heat pump energy use for day t		kWh	
η	gas needed per degree day		m^3/dd	0.39
ϵ	electricity needed per degree day		kWh/dd	4.0
π_{gas}	price of gas		$euro/m^3$	0.8
π_{el}	price of electricity		$euro/kWh$	0.16

The model as described by (1), (2), and (3) above has been used to perform a number of simulations over 2012, thereby using the temperature data from Section 3 and the linear approximation of SPF shown in Fig. 3. Here for each day, depending on the outdoor temperature, a choice was made by the allocation agent to allocate either the gas-based agent or the heatpump-based agent to the heating task. The choice criterion concerns how the outdoor temperature $T_{od}(t)$ compares to some fixed *threshold value* T_{th} for the outdoor temperature:

$$\begin{aligned}
 T_{od}(t) \geq T_{th} &\Rightarrow \text{allocation to heatpump agent} \\
 T_{od}(t) < T_{th} &\Rightarrow \text{allocation to gas-based heating agent}
 \end{aligned}$$

First, in Fig. 7 results from a simulation are shown in which the allocation agent has allocated the gas-based agent to all days with average outdoor temperature below $T_{th} = +6^{\circ}C$ and the heatpump-based agent for the other days. In this case the overall costs are € 388 for gas and € 260 for electricity for the heatpump (€ 648 in total).

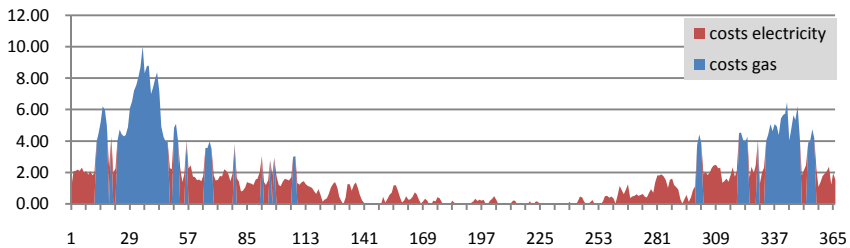


Fig. 7. Example simulation with allocation threshold temperature $+6^{\circ}C$. Overall gas cost € 388, overall heatpump cost € 260, total costs € 648.

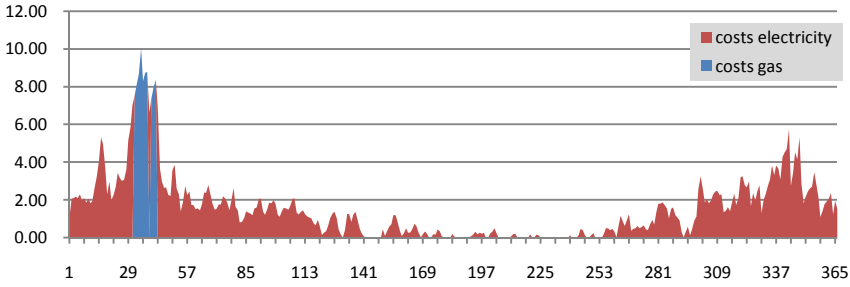


Fig. 8. Example simulation with allocation threshold temperature -4°C . Overall gas cost € 84, overall heatpump cost € 489, total costs € 573.

Fig. 8 shows a similar simulation, but this time the allocation agent has allocated the gas-based agent to all days with average outdoor temperature below $T_{th} = -4^{\circ}\text{C}$ and the heatpump-based agent for the other days. In this case the gas costs over 2012 are € 84, the electricity costs € 489, in total € 573.

For the extreme cases in which only one agent is used for the whole year the total costs are € 815 (only gas) and € 590 (only heatpump). Note that both are higher than the combined simulation shown in Fig. 8.

An overview of the year costs for several what-if simulations for which the threshold value T_{th} for allocation varied from -13°C to $+12^{\circ}\text{C}$ is shown in Fig. 9 below. It is shown that there is a minimum somewhere between 0°C and -8°C ; this minimum represents the most economic choice for the threshold temperature T_{th} . In the next section, in a more general setting this minimum will be determined more precisely by mathematical analysis.

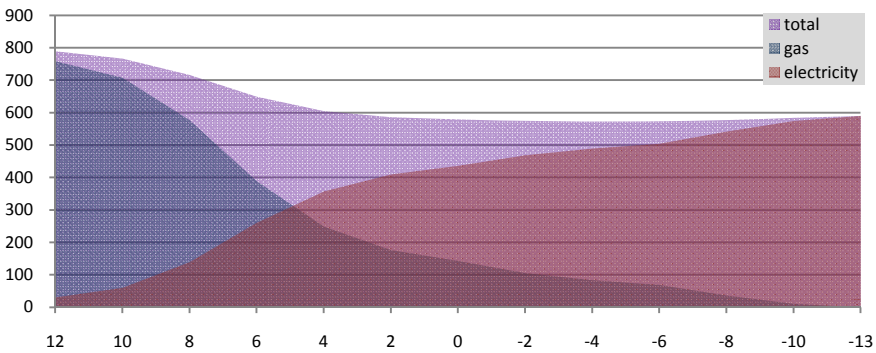


Fig. 9. Overview of costs for different simulations over 2012 for varying allocation threshold temperatures from 12°C to -13°C . The vertical axis represents the costs for the whole year.

This method based on what-if analysis using simulation can be used by the allocation agent HAA in its Agent-Specific Task. However, this makes the model rather complex. In next section, based on mathematical analysis a more compiled form of knowledge will be found that will provide a more concise model for agent HAA.

5 Mathematical Analysis

In this section by mathematical analysis some conclusions are derived from the model introduced in Section 4. These conclusions will provide more adequate knowledge for the allocation agent HAA. First the outdoor temperature threshold value is considered that is used by the allocation agent to decide which agent should get the heating task allocated. It will be determined what this threshold value should be to obtain minimal overall costs (the lowest point in the graph for total costs in Fig. 9). Recall the two expressions (2) and (3) for gas costs $gc(t)$ and heatpump costs $pc(t)$ in Section 4:

$$gc(t) = \pi_{gas} \eta dd(t) \quad (2)$$

$$pc(t) = (\pi_{el} \varepsilon / SPF(T_{od}(t))) dd(t) \quad (3)$$

By comparing these expressions it follows that the following hold:

- Heatpump more economic than gas-based system:
 $\pi_{el} \varepsilon / SPF(T_{od}(t)) < \eta \pi_{gas} \Leftrightarrow SPF(T_{od}(t)) > \pi_{el} \varepsilon / (\eta \pi_{gas})$
- Heatpump and gas-based system equally economic:
 $\pi_{el} \varepsilon / SPF(T_{od}(t)) = \eta \pi_{gas} \Leftrightarrow SPF(T_{od}(t)) = \pi_{el} \varepsilon / (\eta \pi_{gas})$
- Heatpump less economic than gas-based system:
 $\pi_{el} \varepsilon / SPF(T_{od}(t)) > \eta \pi_{gas} \Leftrightarrow SPF(T_{od}(t)) < \pi_{el} \varepsilon / (\eta \pi_{gas})$

It turns out that the value of $\pi_{el} * \varepsilon / (\eta * \pi_{gas})$ serves as a the *optimal threshold value for the performance factor*, denoted by SPF_{oth} :

$$SPF_{oth} = \pi_{el} \varepsilon / (\eta \pi_{gas})$$

For values for π_{el} , ε , η , and π_{gas} as indicated in Table 1 it holds $SPF_{oth} = 2.05$. So, for this case from the mathematical analysis it follows that for the circumstances modelled in the simulations, for a given average outdoor temperature $T_{od}(t)$ at day t the heatpump needs to have an SPF value of 2.05 or more to be at least as (or more) economic compared to a gas-based system. Therefore, a *rule of thumb* can be used stating:

As long as the performance factor SPF does not drop below 2, it is more economic to allocate the heatpump instead of a gas-based installation for the heating system

As SPF (strictly) monotonically depends on T_{od} , associated to SPF_{oth} , (as long as SPF_{oth} is in the range of SPF), there is a (unique) *optimal threshold temperature* T_{oth} for T_{od} such that $SPF(T_{oth}) = SPF_{oth}$. Then it holds:

Heatpump more economic than gas-based $\Leftrightarrow SPF(T_{od}(t)) > SPF_{oth} \Leftrightarrow T_{od}(t) > T_{oth}$

Heatpump and gas-based equally economic $\Leftrightarrow SPF(T_{od}(t)) = SPF_{oth} \Leftrightarrow T_{od}(t) = T_{oth}$

Heatpump less economic than gas-based $\Leftrightarrow SPF(T_{od}(t)) < SPF_{oth} \Leftrightarrow T_{od}(t) < T_{oth}$

Assuming the linear approximation of SPF , for $T_{water} = 50^\circ\text{C}$ in the following way the optimal threshold value T_{oth} can be expressed in $SPF(T_{oth})$:

$$SPF(T_{oth}) = 7.5 - 0.1 * (50 - T_{oth}) \quad \Leftrightarrow$$

$$\begin{aligned}
 SPF(T_{oth}) &= 7.5 - 0.1*(50 - T_{oth}) && \Leftrightarrow \\
 SPF(T_{oth}) &= 2.5 + 0.1 T_{oth} && \Leftrightarrow \\
 T_{oth} &= 10 SPF(T_{oth}) - 25
 \end{aligned}$$

This relation can be used as a form of compiled knowledge by the Agent Specific Task component of the Heating Allocation Agent HAA. For values for π_{el} , ε , η , and π_{gas} as indicated in Table 1 it holds $SPF(T_{oth}) = SPF_{oth} = \pi_{el}\varepsilon / (\eta \pi_{gas})$; therefore:

$$T_{oth} = 10 * 2.05 - 25 = -4.5^\circ\text{C}$$

Similarly assuming the quadratic approximation of SPF , for $T_{water} = 50^\circ\text{C}$ the value of T_{oth} can be determined:

$$\begin{aligned}
 SPF(T_{oth}) &= 7.45 - 0.1*(50 - T_{oth}) + 0.004 T_{oth}^2 && \Leftrightarrow \\
 0 &= 2.45 - SPF(T_{oth}) + 0.1 T_{th} + 0.004 T_{oth}^2 && \Leftrightarrow \\
 T_{oth}^2 + 25 T_{oth} + (2450 - 1000 SPF(T_{oth}))/4 &= 0 && \Leftrightarrow \\
 T_{oth} &= (-25 +/- \sqrt{25^2 - 2450 + 1000 SPF(T_{oth})})/2 && \Leftrightarrow \\
 T_{oth} &= (-25 +/- \sqrt{1000 SPF(T_{oth}) - 1825})/2
 \end{aligned}$$

For the values from Table 1 it holds $SPF(T_{oth}) = SPF_{oth} = \pi_{el} \varepsilon / (\eta \pi_{gas}) = 2.05$. Then this becomes

$$\begin{aligned}
 T_{oth} &= (-25 +/- \sqrt{225})/2 = (-25 +/- 15)/2 \\
 T_{oth} &= -20 \text{ or } -5
 \end{aligned}$$

As -20 falls outside the range from -10 to 10 of the approximation, the only relevant value is $T_{oth} = -5^\circ\text{C}$. This is close to the value -4.5 that came out using the linear approximation of SPF .

The mathematical analysis can also be used to determine T_{water} for which the heatpump is always most economic to allocate, in the following manner. The minimal average day temperature in 2012 was -13°C . Therefore, when SPF is at least $SPF(T_{oth})$ for temperatures as low as -13°C , then the heatpump is always most economic. Then, assuming that the linear approximation for SPF also holds for values of T_{water} lower than 50°C , the value of T_{water} can be determined as follows:

$$\begin{aligned}
 SPF(T_{oth}) &= SPF(-13) = 7.5 - 0.1*(T_{water} - (-13)) && \Leftrightarrow \\
 0.1 * T_{water} &= 7.5 - 1.3 - SPF(T_{oth}) && \Leftrightarrow \\
 0.1 * T_{water} &= 6.2 - SPF(T_{oth}) && \Leftrightarrow \\
 T_{water} &= 62 - 10 SPF(T_{oth}) && \Leftrightarrow
 \end{aligned}$$

For the values from Table 1 it holds $SPF(T_{oth}) = 2.05$; then this becomes

$$T_{water} = 41.5^\circ\text{C}$$

The same can be done using the quadratic approximation of SPF .

$$\begin{aligned}
 SPF(T_{oth}) &= 7.45 - 0.1*(T_{water} - (-13)) + 0.004 (-13)^2 && \Leftrightarrow \\
 SPF(T_{oth}) &= 7.45 - 0.1*T_{water} - 1.3 + 0.004*13^2 && \Leftrightarrow
 \end{aligned}$$

$$\begin{aligned}
 0.1 T_{water} &= 6.15 + 0.004 * 13^2 - SPF(T_{oth}) && \Leftrightarrow \\
 T_{water} &= 61.5 + 6.8 - 10SPF(T_{oth}) && \Leftrightarrow \\
 T_{water} &= 68.3 - 10SPF(T_{oth})
 \end{aligned}$$

For the values from Table 1 it holds $SPF(T_{oth}) = 2.05$; then this becomes

$$T_{water} = 47.8^{\circ}\text{C}$$

So, an estimation based on the linear and quadratic approximation may be that when the water in the heating system is kept around 45°C , then always allocation of the heatpump is more economic. However, the two estimations differ by 6°C , so they may not be very accurate.

6 Discussion

In this paper a multi-agent model for a domestic heating task was introduced and analysed. The aim was to model a hybrid heating system as often used in practice by an agent-based modelling perspective. The model includes two alternative heating agents (for gas-based heating and for heatpump-based heating), and a third allocation agent which determines the most economic allocation of the heating task to these heating agents over the days in a year. To be able to make good allocation decisions first it was analysed how the performance of a heatpump depends on the outdoor temperature. One method discussed was a what-if analysis method using agent-based simulation, and tried out for realistic data from 2012. Another method discussed was by mathematical analysis deriving more precise knowledge about the most optimal allocation choice. These methods can be used by the allocation agent to determine in a dynamic, adaptive manner per day or per hour a most economic allocation, depending on the (predicted) outdoor temperature.

Note that for the sake of simplicity heating costs were assumed proportional to the energy used. In practice, often a distinction is made in fixed costs and variable costs, where fixed costs have to be paid even if no energy is used at all. The methods described in the paper can easily be adapted to include such cost models. Although the paper has focused on optimal decisions from a financial perspective, the methods introduced can also be applied to other aspects, for example, CO_2 emission. It is also easy to extend the model by using more than two heating agents, and to incorporate environmental dynamical models within the allocation agent, for example, to predict the outdoor temperature (e.g., [14]). Another extension of the presented approach is to allow for partial allocation, in which can the allocation is not exclusive, but also have the form of allocation both agents at the same time, but each for a certain fraction of the heating task. Finally, another interesting extension is to analyse the current approach when it is combined with electricity production by solar panels (e.g., [2, 8, 10, 11, 15]).

References

1. Acikcalp, E., Aras, H.: Comparing Geothermal Heat Pump System with Natural Gas Heating System. In: Moshfegh, B. (ed.) Proc. of the World Renewable Energy Congress, Sweden, pp. 1337–1344 (2011)
2. Bakirci, K., Ozyurt, O., Comakli, K., Comakli, O.: Energy analysis of a solar-ground source heat pump system with vertical closed-loop for heating applications. *Energy* 36, 3224–3232 (2011)
3. Bosse, T., Hoogendoorn, M., Klein, M.C.A., Treur, J.: An Ambient Agent Model for Monitoring and Analysing Dynamics of Complex Human Behaviour. *Journal of Ambient Intelligence and Smart Environments* 3, 283–303 (2011)
4. Brazier, F.M.T., Dunin Keplicz, B., Jennings, N., Treur, J.: DESIRE: Modelling Multi-Agent Systems in a Compositional Formal Framework. *International Journal of Cooperative Information Systems* 6, 67–94 (1997)
5. Brazier, F.M.T., Jonker, C.M., Treur, J.: Principles of Component-Based Design of Intelligent Agents. *Data and Knowledge Engineering* 41, 1–28 (2002)
6. Brazier, F.M.T., Jonker, C.M., Treur, J.: Compositional Design and Reuse of a Generic Agent Model. *Applied Artificial Intelligence Journal* 14, 491–538 (2000)
7. Busato, F., Lazzarin, R.M., Noro, M.: Two years of recorded data for a multisource heat pump system: A performance analysis. *Applied Thermal Engineering* 57, 39–47 (2013)
8. Carli, D., Ruggeri, M., Bottarelli, M., Mazzer, M.: Grid-assisted photovoltaic power supply to improve self-sustainability of ground-source heat pump systems. In: Proc. of the 2013 IEEE Int. Conf. on Industrial Technology (ICIT), pp. 1579–1584. IEEE (2013)
9. Chua, K.J., Chou, S.K., Yang, W.M.: Advances in heat pump systems: A review. *Applied Energy* 87, 3611–3624 (2010)
10. Deng, S., Dai, Y.J., Wang, R.Z., Matsuura, T., Yasui, Y.: Comparison study on performance of a hybrid solar-assisted CO₂ heat pump. *Applied Thermal Engineering* 31, 3696–3705 (2011)
11. Deng, S., Dai, Y.J., Wang, R.Z.: Performance study on hybrid solar-assisted CO₂ heat pump system based on the energy balance of net zero energy apartment. *Energy and Buildings* 54, 337–349 (2012)
12. Hepbasli, A., Kalinci, Y.: A review of heat pump water heating systems. *Renewable and Sustainable Energy Reviews* 13, 1211–1229 (2009)
13. Jeong, J., Kim, C.S., Lee, J.: Household electricity and gas consumption for heating homes. *Energy Policy* 39, 2679–2687 (2011)
14. Treur, J., Umair, M.: An Agent Model Integrating an Adaptive Model for Environmental Dynamics. *Int. J. of Intell. Inf. and Database Systems* 5, 201–228 (2011)
15. Xu, B., Zhang, Y., An, J., Han, K., Yi, J.: Feasibility study of the winter heating about solar - water source heat pump system in hot summer and cold winter zone. In: Proc. of the 4th Electronic System-Integration Technology Conference (ESTC 2012), pp. 289–292. IEEE (2013)
16. <http://www.liveheatpump.com>

Improving Human-Aware Planning

Sebastian Ahrndt

DAI-Laboratory of the Technische Universität Berlin
Department of Electrical Engineering and Computer Science
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
`sebastian.ahrndt@dai-labor.de`

Abstract. When it comes to planning for joint human-agent activities one has to consider not only a flexible plan execution and social constraints but also the dynamic nature of humans. This can be done by providing human behavioral models about abilities and preferences and adapting these models to individual preferences during the interaction. The goal of this PhD is to examine whether this combination leads to more efficient human-aware planning components.

1 Introduction

To make autonomous agents effective team-players in joint human-agent activities, planning systems have to tackle the dynamic nature of humans [3,4]. Although humans have the ability to act much better under uncertainty than computers, their behaviour features aspects of uncertainty for the planning process. As an example, humans may change their goals from one moment to another without a (for computers) comprehensible reason. Furthermore, humans may interrupt tasks or execute them in a non-optimal way.¹ Indeed, humans generally select merely good and feasible actions rather than the optimal ones [10]. These aspects of uncertainty affect the search for a feasible sequence of actions in different ways. Where a non-optimal execution might influence only the execution time of a specific task, the sudden interruption of a task endangers the whole plan and therefore the ability to reach a given goal. Hence, one has to consider *that whenever a task is assigned to a human with the ability to fulfill it, the human may accept or decline such task and provide results either in time or delayed* [6]. Such form of ‘context-dependent’ behaviour constitutes the problem addressed by this PhD thesis.

Planning procedures that account for joint human-agent activities are done by Human-Aware Planning (HAP) components. The central assumption of these components is that whenever a task is assigned to a human with the ability to fulfill it, the human will provide results in a timely fashion. This assumption is questionable since context-dependent behaviour of humans is completely ignored. This underlines the necessity for a more general assumption, as the one we have mentioned above.

¹ In our domain, this dynamical behaviour can be seen as some kind of ‘Quality of Service’ a human is able to provide.

In essence, it is our intention to better estimate the costs of human capabilities in specific contexts. This refers to a solution that is able to estimate the achievement of actions assigned to humans. The basic idea is to provide more information about humans by defining a meta-model that is able to capture human abilities and behaviours. The meta-model comprises a set of properties that represent the possible behaviour of a human and a set of capabilities that the human can offer in the particular domain. Furthermore—as each human is different—we want to learn individual preferences and habits during the interaction with the human user and consequently refine the human models. This additional information can be used to produce plans more efficiently, compared to the existing state-of-the-art in HAP, where efficiency addresses several measurable factors, such as the number of replannings or the number of failures that are produced before the first success. Hence, the goal of this doctoral thesis is to validate or disprove the following statement: *‘Providing models of human behaviour and tailoring these models to individual preferences and habits of a human improves the efficiency of human-aware planning components’*.

2 Related Work

Klein et al. [4] present several requirements for joint human-agent teamwork. This includes top-level goals like planning and autonomy technologies that enable collaborative approaches. Furthermore, the work underlines that a basic understanding of human abilities and intentions is required. A couple of research groups present HAP components [1,2,3,6,8] that satisfy the one or the other requirement. Table 1 classifies these works in comparison to the thesis that is outlined in this work. The table emphasises the ability of all above-mentioned approaches to monitor the execution phase, to use social constraints, to use models of human abilities and intentions and whether these models are static or introduce some certain kind of learning.

Table 1. A classification of human-aware planning approaches (+ the approach has this feature, o the approach does not have this feature but it is supported in some (weak) way (e.g. extension), – the approach does not have this feature).

	[6]	[1]	[3]	[2]	[8]	My PhD
Execution Monitoring	–	+	+	+	+	+
Social Constraints	+	+	o	o	–	o
Human Models	o	o	+	o	o	+
Learning	–	o	o	–	–	+

Only the work of *Kirsch et al.* [3] supports such models of human abilities ‘out of the box’. The authors emphasise that their models are only the starting point and that some form of learning is required to adapt to individual preferences.

3 Approach

The basic idea of the approach is to provide more information about humans for the HAP component. In order to do so the human users are modeled as agents and therefore as a component of the computing system. The intended meta-model contains a set of properties that represent the potential behaviour of a human (e.g. probability distribution) and a set of capabilities that the human can offer in the particular domain the (multi-agent) system inhabits. Whereas the set of capabilities is domain-dependent and must be specified for each application, the set of properties is based on the *Myers-Briggs Type Indicator* [7] (MBTI) theory. MBTI is a theory about human personality types and their influences on the the decision-making process of humans. In combination with the BDI model of agency [12] it was already shown that MBTI influences the perception phase and the generation of outputs [9]. The approach transfers this knowledge to HAP by investigating how the combination of BDI and MBTI affects estimations of the achievement of actions assigned to humans. Technically, this means how one can receive a more accurate cost estimation (e.g., expected reward) for a specific capability in a specific context.

In addition, as each human is different, the system needs to refine these properties for each human and revises its plans accordingly. Here, an agent-model is required that is able to learn the characteristics of a specific human and to consequently refine the initial human model. A first step towards this requirement has been done by modifying the agent-model for human-aware planning presented by *Montreuil et al.* [6]. It consists of a set of tuples of actions A and context-dependent costs C^{ctx} . Here, the prior static set of context-dependent costs was replaced with an adaptive component. For that, we introduce the set of contexts CTX an agent can be in and a relation $cost$ between the actions, the context and the costs. The modified model of an agent ag looks as follow:

$$\{A_{ag}, C_{ag}, CTX_{ag}, cost : A_{ag} \times CTX_{ag} \rightarrow C_{ag}\}.$$

At first glance, it seems not to be useful to introduce CTX and a new relation as the original agent-model already represents this relation in the set C^{ctx} . Nevertheless, we decided to introduce the extra set as our approach requires some form of learning. For the learning, we plan to apply reinforcement learning (RL) methods [11], that will especially at the very beginning (according to the learning speed) profit from the domain knowledge of the human agents. Our subject to research is to find a mapping between the $cost$ function and the Q-function used in RL.

A final application will contain human agents (as avatars of the currently available humans), software agents and planning capabilities. If the agent-system commits to a new goal, an agent with the capability to plan will collect all information necessary to plan (i.e., build the domain using the information provided by the other agents) and afterwards trigger the associated agents to execute the produced plan. The execution of actions and the learning process will be a responsibility of the individual agent, to enable the approach to use existing planners. To do so, we integrated a generic planning component into the

agent-framework JIAC V [5] using Planning4J.² Furthermore, we already implemented the domain acquisition process and the possibility to assign the plan to the associated agents.

In order to evaluate our approach it is planned to analyse the efficiency and to compare the results to existing HAP components using the metrics of the *International Planning Competition*.³

4 Contribution

The contribution of this PhD is the possibility to produce plans more efficiently as the plan-generation is not only based on general models of human behaviour but also based on individual preferences of the human user who is interacting with the system. For this, the effect of a combination of human models (embedded into a BDI architecture) and different RL techniques on the efficiency of human-aware planning components is examined. Another contribution is a comparison of different RL techniques such as ϵ -greedy, Regret Minimization and Q-Learning [11] and their applicability to human-aware planning. The main contribution is a system that is able to estimate the achievement of actions assigned to the uncontrollable agent also known as human.

References

1. Alili, S., Warnier, M., Ali, M., Alami, R.: Planning and plan-execution for human-robot cooperative task achievement. In: Proc. of the 19th ICAPS (2009)
2. Cirillo, M., Karlsson, L., Saffiotti, A.: Human-aware task planning: An application to mobile robots. *ACM Trans. Intell. Syst. Technol.* 1(2), 1–26 (2010)
3. Kirsch, A., Kruse, T., Mösenlechner, L.: An integrated planning and learning framework for human-robot interaction. In: Proc. of the 19th ICAPS (2009)
4. Klein, G., Woods, D.D., Bradshaw, J.M., Hoffmann, R.R., Feltovich, P.J.: Ten challenges for making automation a “team player” in joint human-agent activity. *Human-Centered Computing* 19(6), 91–95 (2004)
5. Lützenberger, M., et al.: JIAC V — A MAS framework for industrial applications (extended abstract). In: Proc. of the 12th AAMAS, pp. 1189–1190 (2013)
6. Montreuil, V., Clodic, A., Alami, R.: Planning human centered robot activities. In: Proc. of the ISIC 2007, pp. 2618–2623. IEEE (2007)
7. Myers, I.B., Byers, P.B.: *Gifts Differing: Understanding Personality Type*, 2 edn. Nicholas Brealey Publishing (May 1995)
8. Rosenthal, S., Biswas, J., Veloso, M.: An effective personal mobile robot agent through symbiotic human-robot interaction. In: Proc. of the 9th AAMAS (2010)
9. Salvit, J., Sklar, E.: Modulating agent behavior using human personality type. In: Proc. of the WS on HAIDM at 11th AAMAS, pp. 145–160 (2012)
10. Simon, H.A.: *The Sciences of the Artificial*, 3rd edn. MIT Press (October 1996)
11. Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Intro*. MIT Press (1998)
12. Wooldridge, M.: *Reasoning about Rational Agents*. Intelligent Robotics and Autonomous Agents. The MIT Press (July 2000)

² Planning4J – <https://code.google.com/p/planning4j/>

³ International Planning Competition – <http://ipc.icaps-conference.org/>

Best First Search Planning of Service Composition Using Incrementally Refined Context-Dependent Heuristics

Johannes Fähndrich

DAI-Labor, Technische Universität Berlin
johannes.fahndrich@dai-labor.de

Abstract. In order to decide if a agent capability is helpful to achieve a goal, modern search algorithms in AI research use heuristics to narrow the search space by indicating which capability is the best to use. Considering the lack of information about pragmatic meaning, creating sound heuristics automatically out of capability descriptions asks too much of modern reasoning algorithms. Most approaches use semantics in order to enable the reasoner to improve Word-sense disambiguation in their ontology matching tasks. As semantics are meant to be shared, the information is context independent and quite general. I postulate that context-dependent meaning can play an important role in describing the meaning of concepts used, as some meaning might change with the changes in context. The proposed thesis creates context-dependent heuristics by combining expert knowledge with machine learning. The PhD has the goal of structuring descriptions with a concept introduced in linguistics, introducing a description of domain knowledge and contextual information and thereby enable the automatic creation of context-dependent heuristics. Choosing from the many improvement points of agent planning, this work focuses on the improvement of capability descriptions.

Introduction. To emerge new capabilities in an autonomous manner, agent planners explore a search space of capabilities and compose them into a plan. Since this search space might become large, sometimes infinite, heuristics are used to decide which part of the search space to explore. A learning planner has to calculate heuristics on the information provided by the description of a capability. Thus, making such descriptions a central part of an efficient search of capabilities during planning.

The proposed research, argues that for a reasoner to be able to create sound heuristics, context-dependent meaning has to be formulated. Thus, the presented approach describes meaning in a context-dependent manner to extend the traditional semantic description of meaning e.g. introduced by [1,12,13,14] to a pragmatic¹ one. Due to missing context information such semantic information is rather general (context independent) [11]. The approached problem here is the mediocre performance of AI algorithms using capability descriptions like service matchers and agent planners [10,15].

¹ Pragmatic is defined as context-dependent meaning [11].

In order to improve such capability descriptions, the goal of my PhD is to investigate the use of a concept introduced in linguistics as a mechanism to structuring semantic and contextual information using a meta-model called **Natural Semantic Metalanguage** (NSM) [8].

The problem at hand is the extraction of sound heuristics out of capability descriptions. The proposed thesis will answer the following questions:

1. How can meaning be described in a context-dependent manner?
2. How can context-dependent heuristics be created by observing capability descriptions?
3. How to improve performance of agent planners by using context-dependent heuristics?

The indispensable extensions of a description language (e.g. OWL2) with concepts from the NSM raise new challenges for artificial reasoners leading to new reasoning algorithms proposed. The results will be evaluated through the precision and recall values of state-of-the-art service matcher and AI planner.

Approach. We approach the problem of describing context-dependent meaning by using NSM expressions with a domain and context-dependent rating. The NSM theory states that every natural language has a semantic crux consisting of semantic primes and that with those semantic primes the meaning of every concept in that language can be explained [8]. The meaning of concepts is broken down in a decompositional manner, until the expressions consists solely of the semantic primes. A schema of the decomposition into NSM semantic primes is shown in figure 1 which presents the approach to the first research question.

The semantic primes are regarded as atomic building blocks of meaning, which means they do not need any further explanation. In AI this can be formulated as a part of the meta language of the reasoner, extending a general ontology language like OWL2 with new concepts. My research provides an extension of the work of *Bouquet* [2], *Giunchiglia* [7] and *Ghidini et al* [6] who contextualize ontologies.

The meaning is then enriched with a domain specific rating. The rating is based on three NSM primes: 'GOOD', 'BAD' and 'VERY' creating five² classes of rating. For instance, the concept 'unlocked' might be assigned to 'unsecure' in a security domain. Depending the context a bathroom door might be 'GOOD' to be unlocked, in contrast to a bank vault.

A learning algorithm exploiting such a description to create a context-dependent heuristic is the essence of the proposed research. The first step towards a formal framework including a measure for the degree of self-explaining descriptions have been published [4]. To answer the second research question a reasoner is adapted to create heuristics based on the descriptions utilizing NSM expressions. These heuristics will be simple at first, i.e. a count of 'GOOD' vs. 'BAD', and will be developed towards more sophisticated ones.

² Reaching from 'VERY BAD' to 'VERY GOOD', including no rating as neutral.

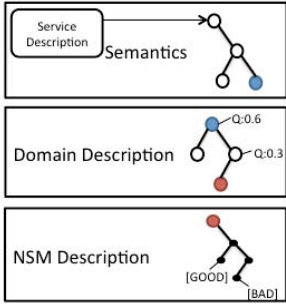


Fig. 1. Decomposition of meaning into NSM Primes

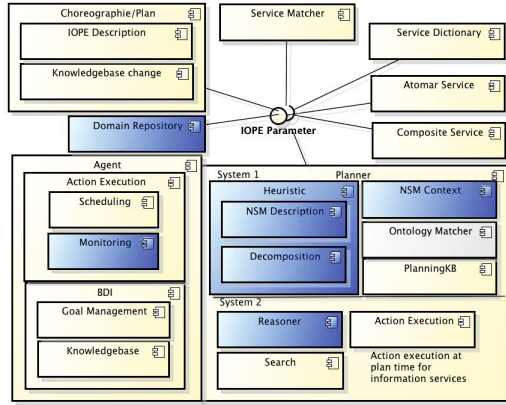


Fig. 2. Component diagram of the extended AI planner

In order to evaluate the heuristics we extend a planner with a learning component, which will add context-dependent heuristics to the action description by learning the quality of the executed plan. In more detail, we extend a Hierarchical Task Network (HTN) planner as shown in Figure 2.

In Figure 2 the emphasized elements are components that will be modified during the implementation of this work. In the extended AI planner, the *SeMa*² [10] is used as a Service Matcher to match capabilities with required subgoals. The *Domain Repository* holds the domain specific descriptions of the semantic concepts. The *NSM Context* component holds the context-dependent descriptions of meaning, which can be decomposed into the NSM concepts that hold domain specific meanings made available through the *Domain Repository*. The *monitoring* component measures the quality of the plan/orchestration and enables the heuristic to adapt to changes. The output of the monitoring is used as feedback for the learning input (reinforcement). A truncated multi-step Q-learning algorithm [3] is used to learn a heuristics function $Q : A \times S \rightarrow \mathbb{R}$, where the action A is an applicable action and the system state S consists of the set of all preconditions and effects. The *Heuristic* component *decomposes* the capability description and enriches them with *NSM Descriptions* including the action-value-function learned which adapts the context-dependent meaning and with that create sound heuristics, evaluating the third research question.

Contribution. The contribution of the proposed thesis is based on the analysis of the benefit of NSM as a semantic meta-model for the generalize over capability descriptions. The thesis outlines an approach able to create NSM-based explanations that can be used by artificial reasoners to search on them. This search is guided by context-dependent heuristics that will be provided by a learning component monitoring the plan execution. The extension of semantic meaning to a context-dependent meaning is argued. Further the proposed thesis tackles the problem of

evaluating high level descriptions of capabilities in regard to a given goal (producing a context-dependent heuristic). One outcome is a method of describing the context-dependent meaning. Another outcome is the extension of a planning system with a learning component that will learn context-dependent heuristics given NSM descriptions which is one step towards filling the performance gap between static and learning planners [5].

Evaluation. Since it is the goal of this research to foster context dependent heuristics created from capability descriptions, the evaluation of such heuristics will be a quantitative analysis of the created learning planner. As a evaluation goal, the 1080 services and appropriate queries in the S3 contest will be described using the NSM approach. Established measures of success and planner performance are use i.e. *percent of successful plans*, *average number of capabilities use* and the *average time to create a successful plan* [16] for the plan evaluation. Further a more theoretical evaluation will be based on *Kaddoum* [9] and elicit the effect of the self-explanatory description on the adaptive system using them.

References

1. Bencomo, N., Cavallaro, L., Sawyer, P., Sykes, D., Issarny, V.: Satisfying requirements for pervasive service compositions. ACM, Innsbruck (2012)
2. Bouquet, P., Giunchiglia, F.: C-owl: Contextualizing ontologies (2003)
3. Chen, S., Wu, H., Han, X., Xiao, L.: Multi-step truncated q-learning algorithm. *Machine Learning*, 18–21 (2005)
4. Fährndrich, J., Ahrndt, S.: Towards self-explaining agents. In: PAAMS 2013 (2013)
5. Fern, A., Khardon, R., Tadepalli, P.: The first learning track of the international planning competition. *Machine Learning* 84(1-2), 81–107 (2011)
6. Ghidini, C., Giunchiglia, F.: Local models semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 1–41 (2001)
7. Giunchiglia, Maltese, Dutta: Domains and contest: Fiest steper towards managing diversitz in knowledge. *Journal fo Web Semantics* 12-13, 53–63 (2012)
8. Goddard, C. (ed.): *Cross-linguistic semantics*. John Benjamins Pub. Co. (2008)
9. Kaddoum, G., George, P.: Characterizing and evaluating problem solving self-* systems. In: *Future Computing*, pp. 137–145 (2009)
10. Klusch, M., Küster, U., Leger, A., Martin, D., Paolucci, M.: S3 (2012), <http://www-ags.dfki.uni-sb.de/~klusch/s3/s3c-2012-summary-report.pdf> (last visited: June 05, 2013)
11. Löbner, S.: *Semantik. Eine Einführung* (2003)
12. Martin, D., et al.: Bringing semantics to web services: The OWL-S approach. In: Cardoso, J., Sheth, A.P. (eds.) *SWSWPC 2004*. LNCS, vol. 3387, pp. 26–42. Springer, Heidelberg (2005)
13. Oaks, P., ter Hofstede, A.H.M., Edmond, D.: Capabilities: Describing what services can do. In: Orłowska, M.E., Weerawarana, S., Papazoglou, M.P., Yang, J. (eds.) *IC-SOC 2003*. LNCS, vol. 2910, pp. 1–16. Springer, Heidelberg (2003)
14. Polleres, A., Lausen, H., Lara, R.: Semantische beschreibung von web services. In: Pellegrini, T., Blumauer, A. (eds.) *Semantic Web*, pp. 505–524. Springer (2006)
15. Tamani, E., Evripidou, P.: Combining pragmatics and intelligence in semantic web service discovery. In: Meersman, R., Tari, Z. (eds.) *OTM-WS 2007, Part II*. LNCS, vol. 4806, pp. 824–833. Springer, Heidelberg (2007)
16. Yoon, S.W., Fern, A., Givan, R.: Ff-replan: A baseline for probabilistic planning. In: *ICAPS*, p. 352. AAAI (2007)

Sequential Single-Cluster Auctions

Bradford Heap

School of Computer Science and Engineering
The University of New South Wales
Sydney, NSW, 2052, Australia
bradfordh@cse.unsw.edu.au

1 Introduction

My thesis is concerned with task allocation in multi-robot teams operating in dynamic environments. The key contribution of this work is the development of a distributed multi-robot task allocation auction that allocates clusters of tasks to robots over multiple bidding rounds. Empirical evaluation has shown this auction routine performs well in handling online task insertion and task reallocation upon robot failure.

The *multi-robot task allocation* (MRTA) problem is a complex NP-Complete optimisation problem with globally optimal solutions that are often difficult to find. Alternatively, rapid generation of near optimal solutions that minimise task execution time and/or energy used by robots are often highly desired. My approach seeks to cluster together closely related tasks and then builds upon existing distributed market-based auction architectures for distributing these sets of tasks among autonomous robots.

Dynamic environments introduce many challenges that are not found in closed systems. For instance, it is common for additional tasks to be inserted into a system after an initial solution is determined. Additionally, it is highly likely in long-term autonomous systems that individual robots may suffer some form of failure. The ability to alter plans in order to react to these types of challenges in a dynamic environment is required for the completion of all tasks. In my approach I allow the repeated formation and auctioning of task clusters with varying number of tasks. This allows us to react to and change the task allocation among robots during execution.

Throughout my research I use empirical evaluation to study different approaches for forming clusters of tasks and the application of task clustering to distributed auctions for MRTA problems. The results show that allocating clusters of tasks to robots in solving these types of problems is a fast and effective method and produces near optimal solutions.

2 Related Work

Market-based distributed auction algorithms are popular in the robotics community for solving static MRTA problems [1,6]. In particular, *sequential single-item auctions* (SSI auctions) which allocate tasks over multiple rounds [7] are

well studied. Although, SSI auctions produce team costs that are generally sub-optimal, they have low communication and winner determination costs which result in a quick allocation of tasks. A variety of improvements and extensions to SSI auctions have been studied which trade-off allocation time against overall team costs [6].

Despite these improvements, a major computational challenge remains in the performance of auction mechanisms that consider inter-task synergies.¹ In many auction algorithms, increasing the number of tasks causes a combinatorial explosion in the number of calculations required to form task bids. Compounding this further, as the number of robots increases, the communication and computational requirements for winner determination also increase. As a result the suitability of these techniques in large real-world scenarios is limited. Forming clusters of tasks has been explored by a number of researchers as a method to reduce the combinatorial explosion of increasing task counts [8,9,10].

3 Sequential Single-Cluster Auctions

For my first major piece of work, I expanded upon SSI auctions to develop *sequential single-cluster auctions* (SSC auctions) [2]. In SSC auctions a clustering algorithm forms fixed clusters of geographically close tasks which robots subsequently bid on using an SSI-like auction technique. Auctioning clusters of tasks reduces the numbers of bids required and thus reduces the communication overhead. Experimental results show that, on average, SSC auctions produce lower cost solutions than standard SSI auctions. Furthermore I expanded upon this work by repeatedly generating clusters of uncompleted tasks and auctioning them [4]. This extension demonstrated that repeatedly forming clusters with different task memberships allows robots to consider many combinations of inter-task synergies that are not considered during SSI auctions which only allocate tasks once.

Initially, I used K -means clustering to form clusters of tasks for auction to robots. In the next piece of work I compared *centroid-based clustering* (K -means and K -medoids clustering) to *agglomerative clustering* (single-linkage and complete-linkage clustering). I also considered the differences between straight line distance and true path distance (which takes into consideration obstacles between tasks) as metrics in cluster formation and priority allocation of clusters with many tasks. Overall, empirical results show agglomerative clustering with a true path distance metric when used in SSC auctions produces low cost solutions to MRTA problems. Analysis of the time required to form clusters shows that using a true path distance metric is around 100 times slower than using straight line distances. Additionally, when minimising the total execution time of a system, the priority allocation of large task clusters was beneficial. Part of this work, comparing K -means clustering to single-linkage clustering was published in late 2012 [3].

¹ An inter-task synergy is where the cost to complete two tasks in unison is cheaper than completing each task individually.

4 Task Allocation with Pickup and Delivery in Dynamic Environments

In my most recent work I consider an extension of the standard MRTA problem for robots with tasks requiring pickup and delivery in a dynamic domain. To apply SSC auctions to this environment I first had to develop a clustering approach that handles tasks with both pickup and delivery. Standard single-point location based clustering algorithms cannot be directly used as both the starting and terminating points of the tasks need to be considered. To overcome this, I developed a two step clustering approach that first clusters tasks by pickup location and then, within each pickup location task cluster, clusters are formed based on delivery location. This approach allows standard clustering algorithms to be used and is reasonably simple to implement.

Next, I considered the two dynamic scenarios requiring task reallocation: dynamic task insertion and handling robot failure. In both scenarios we evaluate the performance and execution trade-off between reallocating a subset of the total tasks and all uncompleted tasks. As both scenarios require individual robots to commit to additional tasks, the problems of inserting tasks and removing robots superficially appear to be equivalent problems. However, my empirical results for these two problems differ substantially.

Firstly, for dynamic task insertion I compared local replanning versus global reallocation. In local replanning, when a robot is dynamically assigned a new task, the robot replans its path to complete all uncompleted tasks. In global reallocation, when a robot is dynamically assigned a new task it signals to all other robots to begin an auction of all uncompleted tasks across all robots. I compared three different ratios of dynamic to static tasks, 25%, 50%, and 75% unknown at the start to a baseline of all static tasks. Local replanning generally produced the best results when only 25% of tasks are unknown at the start. Contrarily global reallocation performed well in the highly dynamic environment of 75% of tasks being dynamically inserted. I speculate that the reason for this is due to the high numbers of repeated cluster formations exposing many different inter-task synergies during each repeated auction. In the worst-case scenario the maximum distance result was twice that of the corresponding baseline result [5].

Secondly, in handling robot failure I consider the reallocation of a failed robot's assigned tasks to the remaining operating robots. The problem of robot failure is an important consideration in the successful completion of a set of tasks in a distributed robotic system. For example, during task execution individual robots may fail due to malfunctioning equipment or running low on batteries. In this scenario, I consider two different approaches for the reallocation of tasks among the remaining operating robots. *Partial reallocation* in which only the failed robot's uncompleted tasks are auctioned. This results in the remaining operating robots modifying their existing task execution plans to incorporate additional tasks. In contrast to a *global reallocation* of the failed robot's uncompleted tasks and all uninitialised tasks across all operating robots. This is an extension of the approach used for task insertion. While a global reallocation has greater computation and communication needs, more robot/task and inter-task

synergies are considered and therefore it can be expected that this approach would produce lower team costs. However, surprisingly our empirical results show that partial allocations produce final results that on average are equivalent to the results for global reallocation. This result is surprising and contradicts the results for dynamic task insertion. To understand the differences in these dynamic scenarios I generated two simple scenarios in which partial reallocation produces lower cost solutions than global reallocation. This has highlighted a key difference between these two scenarios; in dynamic task insertion, tasks are randomly located whereas in robot failure, tasks are generally located in a geographically close area. This problem continues to be a work in progress.

5 Future Work

To date all empirical evaluations have been simulation based and not validated on physical robot systems. Although simulation is common in many multi-robot auction systems, validation on real robots is highly desired. A possible suitable platform for such experimentation is the newly formed RoboCup Logistics League. Additionally, empirical evaluation with teams of heterogeneous robots and tasks would be of interest. Tasks requiring specific robot capabilities would require appropriate changes to cluster formation algorithms and this would result in disparities in bidding and task allocations based on the abilities of individual robots to complete specific tasks, rather than simply on their location and current task commitments.

References

1. Dias, M.B., Zlot, R., Kalra, N., Stentz, A.: Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE* 94(7), 1257–1270 (2006)
2. Heap, B., Pagnucco, M.: Sequential single-cluster auctions for robot task allocation. In: Wang, D., Reynolds, M. (eds.) *AI 2011. LNCS*, vol. 7106, pp. 412–421. Springer, Heidelberg (2011)
3. Heap, B., Pagnucco, M.: Analysis of cluster formation techniques for multi-robot task allocation using sequential single-cluster auctions. In: Thielscher, M., Zhang, D. (eds.) *AI 2012. LNCS*, vol. 7691, pp. 839–850. Springer, Heidelberg (2012)
4. Heap, B., Pagnucco, M.: Repeated sequential auctions with dynamic task clusters. In: *AAAI* (2012)
5. Heap, B., Pagnucco, M.: Repeated sequential single-cluster auctions with dynamic tasks for multi-robot task allocation with pickup and delivery. In: Klusch, M., Thimm, M., Paprzycki, M. (eds.) *MATES 2013. LNCS (LNAI)*, vol. 8076, pp. 87–100. Springer, Heidelberg (2013)
6. Koenig, S., Keskinocak, P., Tovey, C.A.: Progress on agent coordination with cooperative auctions. In: *AAAI* (2010)
7. Lagoudakis, M.G., Markakis, E., Kempe, D., et al.: Auction-based multi-robot routing. In: *RSS*, pp. 343–350 (2005)
8. Sandholm, T.: Contract types for satisficing task allocation: I theoretical results. In: *Proceedings of the AAAI Spring Symposium: Satisficing Models*, pp. 68–75 (1998)
9. Sariel, S., Balch, T.R.: Efficient bids on task allocation for multi-robot exploration. In: *FLAIRS Conference*, pp. 116–121 (2006)
10. Zlot, R., Stentz, A.: Complex task allocation for multiple robots. In: *ICRA*, pp. 1515–1522 (2005)

Decentralised Collision Avoidance in a Semi-collaborative Multi-agent System

Sascha Hornauer

Carl-von-Ossietzky University of Oldenburg, Germany
sascha.hornauer@uni-oldenburg.de

A new approach is considered which explores a decentralised trajectory optimisation algorithm in partly collaborative multi-agent systems to improve safety and provide reliable collision avoidance for vessels in narrow waterways and the open sea. This research will explore trajectory planning under the hypothesis that not all vessels in an encounter will be able or willing to use the same, proposed system. Planning realistic trajectories, which minimise the need to re-plan, will be achieved by observing the predicted behaviour of uncooperative vessels, based on probabilistic models derived from historic data. Among vessels, which are actively contributing as suggested, trajectories will be optimised, distributed and negotiated. As a result, vessels will be capable of carrying out a manoeuvre, automatically agreed upon by an assistance system, while predicting the reaction of uncooperative vessels thus substantiating the judgement of seafaring personnel.

1 Introduction

Currently, ship collisions still occur on a regular basis due to insufficient information of the crew or the inability to perform the necessary manoeuvres to avoid a collision [1]. The International Regulations for Preventing Collisions at Sea (COLREGs) state in a more general way the mandatory actions to take in order to avoid a collision, but not exactly in terms of precise trajectories. Consequently, there has been extensive research on optimising trajectories which are COLREG compliant, safe and economic [2], [3].

While several collision avoidance methods, based on current information on the bridge, have been investigated to determine the method itself and the moment when it should be executed ([4], [5]), one important practical limitation is often neglected: Even though new expert- and enhanced autopilot-systems are being developed to implement collision avoidance strategies it is improbable that all types of vessels will be equipped with the same system at the same time, or at all, due to cost, space and weight limitations as well as varying acceptance among mariners. Therefore, the procedure for finding optimal routes itself has to take vessels into account which can not contribute towards collision avoidance in the same way. These vessels, which are considered *passive*, have to be modelled respecting the probabilistic nature of their behaviour for an observer with no exhaustive information.

Historic information about the overall long term routes, the immediate short term trajectories and other features of vessels are available in abundance from the collection of data, sent by the Automatic Identification System (AIS), installed on many vessels above a certain size. However, it has been rarely considered to model vessels based on these information to improve collision avoidance methods.

Therefore, a method is proposed to generate optimised trajectories which are the result of a negotiation between actively planning vessels, while including uncooperative, passive vessel whose behaviour will be predicted based on historic probabilistic models. The negotiation and application of the trajectories will be evaluated in a marine simulation to assess the quality of the approach.

2 Collision Avoidance under AIS-Based Prediction of Passive Vessel

In order to avoid collisions and minimise the risk involved as well as the used resources in terms of time and energy, optimal trajectories of vessels in an encounter situation have to be found where the optimum is defined as the shortest and safest set of reciprocal trajectories. Vessels will be defined as *active*, if they are able to contribute to the planning procedure towards a common solution by communicating their intended routes, opposed to *passive* vessels, which do not contribute for arbitrary reasons.

While the physical model will be the same for active and passive vessels, the latter's sequence of steering decisions, controlling the physical model, is based on a Bayesian Model learned from historic AIS-data. The behaviour of a ship is a combination of the physical behaviour of the vessel and the steering decisions of the crew. Even though physical models of a vessel's dynamic have been created and examined, few researchers attempted to model cognitive processing activities that take place when decisions and actions are carried out on a vessel's bridge. [6] developed a highly simplified model of collision avoidance manoeuvres in an electronic warfare environment using the cognitive architecture SOAR. In the German project DGON Bridge [7] developed an initial model of decision making of nautical officers (Nautik PSI) based on the psychological theory PSI. Since it is currently, due to the lack of research in the field, uncertain which factors contribute to form steering decisions, the model will be based on a minimal set of statistically significant factors, based on a variant of the Bayesian Information Criterion as investigated by [8]. This way an approximation of the behaviour of all vessels in a set of data will be derived while the reason, motivation and the involved cognitive processes will not be the subject of this study.

Historic AIS-Information have to be processed to learn the Bayesian model, distinct by vessel-type and geographic location. Due to the International Convention for the Safety of Life at Sea (SOLAS), AIS-receiver or transponder are nowadays mandatory on all vessels which meet certain criteria. Consequently, huge databases can be build containing detailed information about several features of vessels in an area, i.e. the position, true heading, speed over ground and rate of turn. In similar work, AIS-Data was used to calculate traffic models of ships in an area and predict the motion of individual vessels [9], [10].

3 Trajectory Generation and Optimisation

The generation of new trajectories will be realised using a heuristic approach, where trajectories are randomly generated and improved using a genetic algorithm as investigated by [12] and Szlapczynski and Szlapczynska (2011) [13]. This approach will build upon their methodology and adapt their definition of a trajectory: In their work, a trajectory is a sequence of nodes, each containing a position as geographical coordinates, and the speed on the segment between the current and the next node. Initially, a segment with only two nodes, the start and the beginning, is created between the ship and its short term destination. Szlapczynski and Szlapczynska optimise trajectories using genetic operations as mutation, crossover and special parameters, designed in their work to handle the crossing of landmasses more purposefully. The concluded trajectories are finally assessed using a fitness function. Operators which change the initially created segment include the insertion of new nodes, thus creating additional segments in a trajectory, deleting or joining nodes and mutating nodes or whole segments, according to their defined rules. In many generations of the genetic algorithm the trajectories can be gradually improved in this way to lie within constraints, i.e. not to cross a landmass, to obey COLREGs and avoid collision with other ships while minimising the resources needed. Even though it seems plausible, to start the optimisation process from a calculated valid trajectory, Szlapczynski and Szlapczynska found the positive effect of a pre-optimisation with other analytical methods to be negligible while it increased the computation time considerably. However, while the approach by Szlapczynski and Szlapczynska plans all trajectories of all ships in an area, in this paper the approach will only be used to optimise a smaller part of the set of all trajectories, which are part of the negotiation of the respective vessels. A multi-agent system will be modelled to implement the proposed method of finding trajectories for representations of active and passive vessels. However, practical implications from investigations, i.e. by [11] during a search and rescue drill, show the need for a low rate of interaction between agents. The performance and applicability of the designed negotiation design will be evaluated in a simulation.

4 Example for Vessel Interaction

The proposed procedure should be able to optimise trajectories and supply marines with additional steering options which, without further knowledge about the intentions of other vessels, would be neglected:

For example, in figure 1 a passive vessel A crosses the active planning vessel B while another vessel C tries to cross A. Vessel A and B are initially in a standard "crossing situation", as defined by the COLREGs, where vessel A has to give way to vessel B. From historic information the planning algorithm will assume the vessel is most likely to use a trajectory from A1 to A4 with a probability of $P = 0.999$ and plan an optimal trajectory for vessel B and C accordingly.

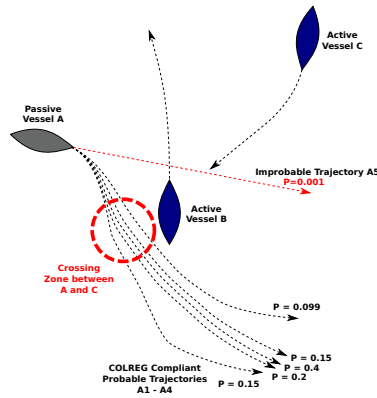


Fig. 1. Three Vessels In A Standard Situation

References

- Hetherington, C., Flin, R., Mearns, K.: Safety in shipping: the human element. *Journal of Safety Research* 37(4), 401–411 (2006)
- Perera, L.P., Carvalho, J.P., Guedes Soares, C.: Fuzzy logic based decision making system for collision avoidance of ocean navigation under critical collision conditions. *Journal of Marine Science and Technology* 16(1), 84–99 (2010)
- Shih, C.H., Huang, P.: Design optimal control of ship maneuver patterns for collision avoidance: a review. *Journal of Marine ...* 20(2), 111–121 (2012)
- Hearn, G.E.: A ship based intelligent anti-collision decision-making support system utilizing trial manoeuvres. In: 2008 Chinese Control and Decision Conference, pp. 3982–3987 (July 2008)
- Tsou, M.: The study of ship collision avoidance route planning by ant colony algorithm. *Journal of Marine Science and Technology* 18(5), 746–756 (2010)
- Kalus, A., Ashford, A., Cook, C.: Cognitive Modelling for Decision Support in Naval Command & Control. *Proceedings of MAICS...*, 25–32 (1999)
- Brüggemann, U., Strohschneider, S., Meck, U.: Virtuelle Nautiker als Probefahrer bei der Neukonzeption von Schiffsbrücken. *Künstliche Intelligenz* (2008)
- Eilers, M., Möbus, C.: Learning the Relevant Percepts of Modular Hierarchical Bayesian Driver Models using a Bayesian Information Criterion 2 Bayesian Autonomous Driver Mixture-of-Behaviors Models, pp. 1–10 (2011)
- Ristic, B., Scala, B.L., Morelande, M.: Statistical analysis of motion patterns in AIS data: Anomaly detection and motion prediction. In: 2008 11th Fusion, pp. 40–46 (2008)
- Gunnar Aarsæther, K., Moan, T.: Estimating Navigation Patterns from AIS. *Journal of Navigation* 62(4), 587 (2009)
- Porathe, T.: Transmitting intended and suggested routes in ship operations: cognitive off-loading by placing knowledge in the world. *Work: A Journal of Prevention, Assessment and ...* 41, 4873–4878 (2012)
- Roman, Smierzchalski, Z.M.: Path planning in dynamic environments. *Innovations in Robot Mobility and Control*, 135–153 (2005)
- Szlapczynski, R., Szlapczynska, J.: On evolutionary computing in multi-ship trajectory planning. *Applied Intelligence* (September 2011)

Opponent Modeling in RoboCup Soccer Simulation 3D

Asma Sanam Larik

Artificial Intelligence Lab, Institute of Business Administration, Karachi, Pakistan
asma.sanam@khi.iba.edu.pk

1 Problem

Opponent modeling [1] is a research area that focuses on the analysis and interpretation of adversary's actions. This modeling, although, is easy for humans but is a challenging task for autonomous multi-agents [2]. In a multi-agent environment, an opponent typically represents a team of agents who are capable of doing decentralized decision making. Thus, an opponent model consists of a collective set of tactical behaviors, termed as strategy, that are exhibited by agents in a dynamic and uncertain environment. RoboCup Soccer [3] provides such an environment and serves as a test bed for analysis and application of new intelligent algorithms. In the context of RoboCup Soccer Simulation League 2D, opponent modeling has been exhaustively researched and various machine learning techniques have been devised for strategy prediction and identification. However, when it comes to RoboCup Soccer Simulation League 3D the problem is much more sophisticated as identification of basic actions that are being performed by a single agent is itself a complex task. This research aims to develop a framework that takes a team of agents as input, analyzes their set of individual actions as well as coordinated team behavior using a set of pre-defined rules, interprets their strategic pattern and suggests a counter strategy. The framework would involve both offline and online learning. The major benefit of this approach is that if we are able to judge our opponent within the first thirty seconds (or so) then we can apply specific rules for exclusively dealing with it. For example, certain teams make excessive use of kicking the ball while many others have the tendency to dribble the ball to maintain possession of it. If the opponent has strong dribbling skill then our best strategy would be to block the path of the opponent and put most of our players to this job. On the other hand, if the opponent team has a tendency to frequently kick the ball then we would like to keep most of our players around our goal area in order to save the goal (especially if the ball is in our half). Thus, if we are able to classify our opponent based upon certain key attributes then we will be able to devise opponent-specific strategy. The motivation behind this approach is that our team KarachiKoalas [4] is applying this technique and utilizing the features obtained from offline team analysis to build a model of the opponent team within initial seconds of the game.

2 Related Work

Opponent modeling has been exhaustively researched in the domain of RoboCup Soccer Simulation 2D. Numerous techniques have been proposed for creation of

these models [5]. Nakashima et al. [6] presented a method that learns opponent team formation from logfiles using neural networks. Fathzadeh et al. [7] have developed three-tier architecture for opponent behavior classification. Kuhlmann et al. [8] created a coach agent whose purpose is to observe patterns of the opponent team, to advise its players and to identify weaknesses of opponent. Iglesias et al. [9] abstract useful features from log files and then analyze these features to recognize different events. Steffens [10] proposed the use of Case-Based Reasoning in order to predict the future opponent actions. Our proposed approach would differ from the above mentioned techniques as our focus is on RoboCup Soccer Simulation 3D environment. Unlike Simulation 2D, where actions executed by the players are recorded in a log file, Simulation 3D matches' log files only contain raw data such as position of the player and ball and does not contain any high level actions such as kicking, dribbling, etc. Thus, we need to devise our own methods that can help in identifying actions taken by an agent. In addition, we also need to devise a mechanism that transforms actions of individual agents to team strategy and as such helps in identifying our opponent type.

3 Proposed Approach and Methodology

The proposed approach comprises of the following phases as depicted in Fig 1:

3.1 Data Extraction from Logfiles

In order to analyze previous matches' data, we need to parse log files of the matches. A parser has already been written [11] that extract positional information of players as well as ball from the log files. After feature extraction of key attributes, some derived attributes as well as temporal attributes are computed such as ball possessor, distance travelled by player, distance between player and ball, velocity of players etc.

3.2 Behavior Definition and Identification Phase

In this phase we utilize the feature extracted previously and formalize behaviors as an action exhibited by ball possessor; a player maintaining minimum distance with the ball. Several cycles constitute a single behavior and dynamic intervals have been defined. Rules have been constituted to differentiate amongst various behaviors such as Kick, Dribble, Shot to Goal etc.. If two or more rules trigger then we perform dynamic conflict resolution technique. These individual behaviors blend to form team behaviors such as the formation the team is playing with, frequency of kicks, average ball possession by team, etc.

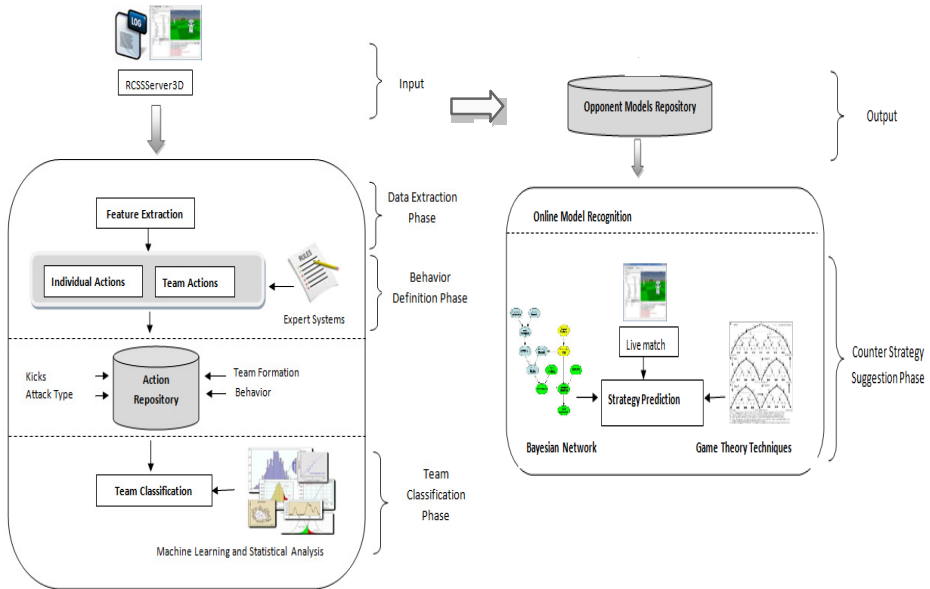


Fig. 1. Proposed Approach

3.3 Team Classification

In this phase we classify teams as strong, medium or weak based upon the features obtained from the initial two phases. We have used the ranking provided by UT Austin Villa 2011 3D simulation team report [12] for distinction between these types.

3.4 Counter Strategy Suggestion

Once an opponent team is classified then we can enable specific rules that can better counter our opponent’s strategy (style of play). The game play would be categorized into set of strategies which would be learnt dynamically by analyzing the specific patterns in the game that signify the attacking style of the opponent. Now this counter strategy would be executed by player nearest to ball or group of players can be advised to take certain positions e.g. to block goal involves group of players while intercepting ball involves just a single player. The novelty of this approach is that we are learning counter strategy against the three categories of the team namely strong, medium and weak. Once team is categorized we further wish to explore the strength/weakness of team. This approach can easily be extended further to categorize amongst groups inside a team such as attackers, defenders etc.

4 Conclusion and Future Work

The presented approach is a methodology for classification of opponent teams based upon their spatial attributes, temporal attributes as well as behavior. We have written

a parser that analyzes a log file and computes individual agent's actions from it and conducted successful preliminary experiments to classify teams as strong, medium and weak team. However, we are still in the process of team strategy identification which is a challenging task. In the future, we wish to extend this approach and try to devise a mechanism that would recommend a counter strategy based upon team classification.

References

1. Borghetti, B.J.: Opponent Modeling in Interesting Adversarial Environments, ProQuest (2008)
2. Bjarnason, R.V., Peterson, T.S.: Multi-agent learning via implicit opponent modeling. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002, vol. 39, pp. 1534–1539. IEEE (2002)
3. Kitano, H., et al.: The RoboCup synthetic agent challenge 97. In: Kitano, H. (ed.) RoboCup 1997. LNCS, vol. 1395, pp. 62–73. Springer, Heidelberg (1998)
4. <http://karachikoalas.iba.edu.pk/>
5. Pourmehr, S., Dadkhah, C.: An Overview on Opponent Modeling in RoboCup Soccer Simulation 2D. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011. LNCS, vol. 7416, pp. 402–414. Springer, Heidelberg (2012)
6. Nakashima, T., Ishibuchi, H.: Mimicking Dribble Trajectories by Neural Networks for RoboCup Soccer Simulation. In: IEEE 22nd International Symposium on Intelligent Control, ISIC 2007, pp. 658–663. IEEE (2007)
7. Fathzadeh, R., Mokhtari, V., Kangavari, M.R.: Opponent Provocation and Behavior Classification: A Machine Learning Approach. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (eds.) RoboCup 2007: Robot Soccer World Cup XI. LNCS (LNAI), vol. 5001, pp. 540–547. Springer, Heidelberg (2008)
8. Stone, P., Kuhlmann, G., Knox, W.B.: Know Thine Enemy: A Champion Robocup Coach agent
9. Iglesias, J.A., Ledezma, A., Sanchis, A.: Caos Coach, Simulation Team: An Opponent Modelling Approach (2006)
10. Steffens, T.: Similarity-based opponent modeling using imperfect domain theories. In: CIG (2005)
11. Larik, A.S., Haider, S.: Rule-Based Behavior Prediction of Opponent Agents Using Robocup 3D Soccer Simulation League Logfiles. In: Iliadis, L., Maglogiannis, I., Papadopoulos, H. (eds.) Artificial Intelligence Applications and Innovations. IFIP AICT, vol. 381, pp. 285–295. Springer, Heidelberg (2012)
12. Patrick, M.A., Urieli, D.: UT Austin Villa. 3D Simulation Team Report (2011)

Policy-Based Approach for Non-Functional Requirements in Self-adaptive and Self-Organizing Systems

Thomas Preisler

Multimedia Systems Laboratory
Faculty of Engineering and Computer Science
Hamburg University of Applied Sciences
Berliner Tor 7, 20099 Hamburg, Germany
thomas.preisler@haw-hamburg.de

Abstract. Today's distributed systems are forced to deal with high and unpredictable dynamics, increasing complexity, and the satisfaction of non-functional requirements (NFR) like, e.g., robustness, availability, and scalability. The research intend presented in this paper aims at providing a policy-based approach to satisfy NFRs in self-governed fashion.

Keywords: Policies, Non-functional Requirements, Self-Organizing.

1 Introduction

Current trends in computer science like mobile and ubiquitous computing in combination with an increasing diversification of hard- and software platforms challenge traditional approaches of engineering and operating distributed systems substantially. Distributed systems have evolved from mainly closed systems with a-priori known tasks, challenges and user to systems with an increasing pervasiveness. These systems have turned into an integral part of the business world as well as the private life of many people. Therefore, this evolution implicates new challenges. The systems are forced to deal with high and unpredictable dynamics, increasing complexity, and the satisfaction of NFR like, e.g., robustness, availability, and scalability. Altogether, this requires a new generation of distributed systems that are capable of adapting their behavior in a self-governed fashion. Self-adaptive and self-organizing (SASO) multi-agent systems (MAS) are a promising approach to deal with this increasing complexity. The research presented in this paper aims at providing an engineering approach as well as a reference architecture to support the satisfaction of NFRs in such systems, based on requirement policies that define conditions for SASO adaptations.

2 Related Work

The challenges identified above are addressed by research areas like Autonomic [2] or Organic Computing [1]. These achieve the SASO properties by introducing different types of feedback loops that are usually implemented by controllers

that form single points of failure. In contrast to feedback loops a policy-based approach for NFRs where the application adapts itself to changing conditions controlled by a centralized policy engine is introduced in [10]. This class of approaches, that introduce centralized control concepts, can be called *self-adaptive systems* [8]. In contrast to this, there are approaches that aim at providing adaptive systems which rely on decentralized control. They are called *self-organizing systems* [8] and seem, due to their decentralized system architecture, to be better suited to deal with the afore-mentioned NFRs. The concept of self-organization has been observed in many other domains like, e.g., biology, physics, or sociology and has, furthermore, proven its applicability for distributed systems already before [3, 9, 12].

3 State of Research

The main challenge in the development of a policy-based approach for the enforcement of NFRs in SASO systems is bridging the semantic gap between the engineering of decentralized self-organization and the semantic description of policies representing the NFRs. In today's complex, distributed IT infrastructures different systems may share the same resources without having any common functional requirements. Thus, a new definition of distributed systems is applicable. Systems that are defined by sharing the same NFRs like robustness or scalability while accessing the same resources in order to fulfill their functional requirements. Such systems have to be capable of adapting their behavior in a self-governed fashion, in order to deal with high and unpredictable dynamics, increasing complexity, and the satisfaction of NFRs. Consequent research questions are (1) concepts development including validation methods, (2) system architecture, and (3) description languages, compilation and middleware support for the engineering of policy-based SASO systems.

The first research question is tackled as part of the case study presented in [4], where a concept was developed to equip a resource-flow system with a self-healing mechanism based on coordination. The NFR of the system was robustness against failures. To ensure these robustness the system was equipped with decentralized coordination processes that allowed it to reconfigure in a self-organizing way. It was realized as a MAS. Simulations of different configurations and system sizes show the scalability and the robustness of the self-healing processes. The systematic validation of self-organizing systems is addressed in [7], where a stochastic validation method based on the microscopic states of the coordination processes under an ergodicity assumption was proposed. Validation methods are crucial for the acceptance of self-organization in mainstream software engineering. The proposed method offers a mesoscopic view by providing information about microscopic characteristics of the coordination process and transition probabilities based on macroscopic system information. It was used to validate the self-healing process of the case study described in [4].

The second research question is tackled by the system architecture proposed in [6]. It emphasizes on a separation of concerns between the coordination logic

and the functional properties of the system, and extends the concept of coordination spaces [11], which provide explicit support for the task of coordination in MAS as part of the agent environment, with a distribution concept to achieve the distribution of coordination information among multiple platforms. As self-organizing applications may undergo internal or environmental conditions spoiling the underlying self-organizing processes, e.g. due to the failure of infrastructure services or inefficient behavior like decrease in performance or starvation, the software architecture is enhanced in [5] with a concept for structural adaptation for self-organizing processes.

Based on this Fig. 1 depicts a conceptual architecture and an engineering approach for policy-based SASO systems as a topic of further research. A system designer specifies NFRs for different systems. A *Transform* component, equipped with different *Language Adapters*, transforms them into machine readable *Requirement Policies*. A system operator with knowledge about the systems infrastructure and resources, specifies *Adaptation Rules* in order to fulfill the NFRs. The transformer also interprets them and generates machine readable adaptation rules. The requirement policies and adaptation rules will be stored in a decentralized *Knowledge and Policy Base*. As the policies have to ensure the satisfaction of the NFRs, they contain conditions for the selection of suitable adaptation rules. These conditions depend on parameter like the system size. It is conceivable that for small system sizes a centralized self-adaptive adaptation rule is more suitable to reconfigure the system in case a NFR is violated. In case of large systems, the centralized solution might not scale well and therefore, a decentralized self-organized adaptation rule is more suitable. The decentralized

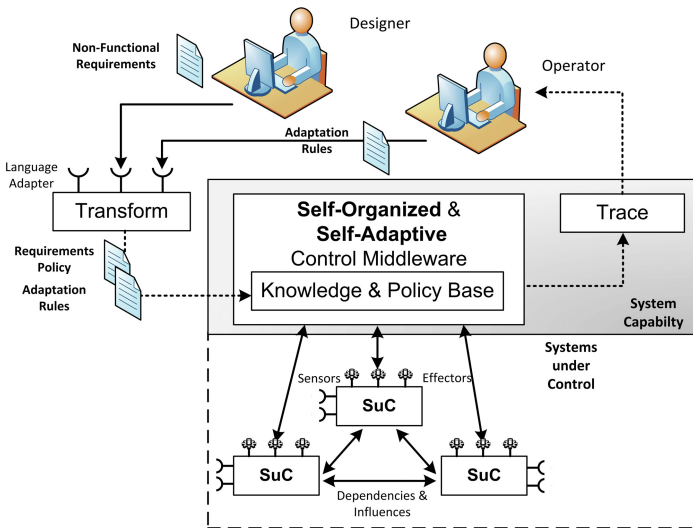


Fig. 1. Conceptual architecture and engineering approach (based on discussion with D. Bade, L. Braubach, A. Pokahr, Hamburg University and W. Renz, HAW Hamburg)

coordination processes examined in the previous research efforts are one possible example for an adaptation rule. The conceived *Control Middleware* selects and executes adaptation rules, based on the policies that ensure the satisfaction of the NFRs. In order to do so, the *Systems under Control* have to be equipped with *Sensors* and *Effectors*. The sensors are conceptualized as decentralized, distributed monitors which supervise the adherence of the policies and in case of a violation select the specified adaptation rule, to reconfigure the systems. The effectors are interfaces used by the control processes to manipulate the system entities in order to ensure the satisfaction of the requirement policies after a violation caused by internal or external events. A distributed *Trace* component allows the operator to track state of the systems and all adaptation efforts undertaken by the middleware. Based on this information the operator is able to comprehend the effects of the adaptation rules and can adjust them manually if necessary.

References

1. Branke, J., Mnif, M., Müller-Schloer, C., Prothmann, H., Richter, U., Rochner, F., Schmeck, H.: Organic computing - addressing complexity by controlled self-organization. In: Proc. of the 2nd Int. Sym. on Leveraging Applications of Formal Methods, Verification and Validation, Washington, DC, USA, pp. 185–191 (2006)
2. Kephart, J., Chess, D.: The vision of autonomic computing. *Computer* 36(1) (2003)
3. Mamei, M., Menezes, R., Tolksdorf, R., Zambonelli, F.: Case studies for self-organization in computer science. *J. Syst. Archit.* 52(8), 443–460 (2006)
4. Preisler, T., Renz, W.: Scalability and robustness analysis of a multi-agent based self-healing resource-flow system. In: Federated Conference on Computer Science and Information Systems, Wroclaw, Poland, pp. 1261–1268 (September 2012)
5. Preisler, T., Renz, W., Vilenica, A.: Structural self-adaptation for self-organizing systems - engineering and evaluation. Submitted to IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems, Philadelphia, USA (September 2013)
6. Preisler, T., Vilenica, A., Renz, W.: Decentralized coordination in self-organizing systems based on peer-to-peer coordination spaces. *ECEASST* 56 (2013)
7. Renz, W., Preisler, T., Sudeikat, J.: Mesoscopic stochastic models for validating self-organizing multi-agent systems. In: IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems Workshops, Lyon, France, pp. 119–126 (September 2012)
8. Salehie, M., Tahvildari, L.: Self-adaptive software: Landscape and research challenges. *ACM Trans. Auton. Adapt. Syst.* 4(2), 14:1–14:42 (2009)
9. Sudeikat, J.: Engineering Self-Organizing Dynamics in Distributed Systems: A Systemic Approach. Dissertation, Universität Hamburg, Fachbereich Informatik, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany (October 2010)
10. Veiga, L., Ferreira, P.: Poliper: policies for mobile and pervasive environments. In: *ARM 2004*, pp. 238–243. ACM, New York (2004)
11. Vilenica, A., Sudeikat, J., Lamersdorf, W., Braubach, L., Pokahr, A.: Coordination in multi-agent systems: A declarative approach using coordination spaces. In: Int. Workshop from Agent Theory to Agent Implementation, pp. 441–446 (2010)
12. Wolf, T.D., Holvoet, T.: A catalogue of decentralised coordination mechanisms for designing self-organising emergent applications. In: Dept. of Comp. Science, K.U. Leuven, Tech. Rep. CW 458. pp. 40–61 (2006)

Author Index

- Ahrndt, Sebastian 400
Apeldoorn, Daan 180
Aschermann, Malte 4
- Basso, Gillian 193
Bazzan, Ana L.C. 73, 292
Becker, Matthias 19
Blatt, Florian 19
Braubach, Lars 29
Bremer, Jörg 208
Briola, Daniela 278
Buccafurri, Francesco 222
Buer, Tobias 166
Burkhard, Hans-Dieter 334
- Comi, Antonello 222
- Davidsson, Paul 101
de Oliveira Ramos, Gabriel 292
Derksen, Christian 236
Dutta, Animesh 278
- Egri, Péter 250
Elakehal, Emad Eldeen 44
- Fähndrich, Johannes 404
Fay, Alexander 236
Fiosina, Jelena 59
Fiosins, Maksims 59
Fortino, Giancarlo 264, 361
- Gaillard, Arnaud 193
Galafassi, Cristiano 73
Ghosh, Supriyo 278
- Heap, Bradford 87, 408
Hilaire, Vincent 193
Hornauer, Sascha 412
- Ivanović, Mirjana 334
- Jander, Kai 29
Jonker, Catholijn 1
- Karänke, Paul 138
Kern-Isberner, Gabriele 124
Klügl, Franziska 101
Koch, Fernando 292
Kopfer, Herbert 166
Koster, Andrew 292
Krümpelmann, Patrick 124
Kułakowski, Konrad 115
- Larik, Asma Sanam 416
Lauri, Fabrice 193
Lax, Gianluca 222
Linnenberg, Tobias 236
- Mariani, Stefano 306
Mascardi, Viviana 278
Matyasik, Piotr 115
Messina, Fabrizio 320
Mitrović, Dejan 334
Montali, Marco 44
Müller, Jörg P. 4, 59
Muscalagiu, Ionel 346
- Negru, Viorel 346
Niazi, Muaz A. 361
- Omicini, Andrea 306
- Padget, Julian 44
Pagnucco, Maurice 87
Paire, Damien 193
Panoiu, Manuela 346
Pappalardo, Giuseppe 320
Pokahr, Alexander 29
Popa, Horia Emil 346
Preisler, Thomas 420
Premm, Marc 138
- Rojas, Raúl 3
Rosaci, Domenico 222, 320
Russo, Wilma 264
- Santoro, Corrado 264, 320
Sarné, Giuseppe M.L. 320
Siddiqa, Amnah 361

Sonnenschein, Michael 208

Szczerbicka, Helena 19

Treur, Jan 372, 386

Unland, Rainer 236

Váncza, József 250

Weiß, Gerhard 2

Widmer, Tobias 138

Yoshida, Norihiko 152

Zhygmanovskyi, Andrii 152

Ziebuhr, Mario 166