

Filip Zavoral
Jason J. Jung
Costin Bădică *Editors*

Intelligent Distributed Computing VII

Proceedings of the 7th International Symposium
on Intelligent Distributed Computing – IDC 2013,
Prague, Czech Republic, September 2013

Studies in Computational Intelligence

Volume 511

Series Editor

Janusz Kacprzyk, Warsaw, Poland

For further volumes:
<http://www.springer.com/series/7092>

Filip Zavoral · Jason J. Jung
Costin Bădică
Editors

Intelligent Distributed Computing VII

Proceedings of the 7th International
Symposium on Intelligent Distributed
Computing – IDC 2013, Prague,
Czech Republic, September 2013

Editors

Filip Zavoral
Faculty of Mathematics and Physics
Charles University in Prague
Prague
Czech Republic

Jason J. Jung
Department of Computer Engineering
Yeungnam University
Gyeongsan
Korea
Republic of (South Korea)

Costin Bădică
Software Engineering Department
Faculty of Automatics, Computers
and Electronics
University of Craiova
Craiova
Romania

ISSN 1860-949X
ISBN 978-3-319-01570-5
DOI 10.1007/978-3-319-01571-2
Springer Cham Heidelberg New York Dordrecht London

ISSN 1860-9503 (electronic)
ISBN 978-3-319-01571-2 (eBook)

Library of Congress Control Number: 2013944764

© Springer International Publishing Switzerland 2014

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.

Printed on acid-free paper

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

Preface

The emergent field of Intelligent Distributed Computing brings together two well established areas of Computer Science: Distributed Computing and Computational Intelligence. Theoretical foundations and practical applications of Intelligent Distributed Computing set the premises for the new generation of intelligent distributed information systems.

Intelligent Distributed Computing – IDC Symposium Series was started as an initiative of research groups from: (i) *Department of Software Engineering, Charles University Prague, Czech Republic* and (ii) *Software Engineering Department of the University of Craiova, Craiova, Romania*. IDC aims at bringing together researchers and practitioners involved in all aspects of Intelligent Distributed Computing. IDC is interested in works that are relevant for both Distributed Computing and Computational Intelligence, with scientific contribution in at least one of these two areas. IDC'2013 was the seventh event in the series and was hosted by *Charles University in Prague, Czech Republic* during September 4–6, 2013. IDC'2013 was collocated with:

- 5th International Workshop on Multi-Agent Systems Technology and Semantics, MASTS'2013;
- 2nd International Workshop on Agents for Cloud, A4C'2013;
- 1st International Workshop on Intelligent Robots, iR'2013.

The material published in this book is divided into five main parts:

- 1 invited contribution;
- 25 contributions of IDC'2013 participants;
- 8 contributions of MASTS'2013 participants;
- 6 contributions of A4C'2013 participants; and
- 2 contributions of iR'2013 participants.

The response to IDC'2013 call for paper was generous. We received 54 submissions from 19 countries. Each submission was carefully reviewed by at least 3 members of the IDC'2013 Program Committee. Acceptance and publication were judged based on the relevance to the symposium themes, clarity of presentation, originality and

accuracy of results and proposed solutions. Finally, 13 regular papers and 12 short papers were selected for presentation and were included in this volume, resulting in acceptance rates of 24 % for regular papers and 22 % for short papers.

Additionally, the co-located workshops (MASTS'2013, A4C'2013, and iR'2013) received 28 submissions. After the careful review (each submission was reviewed by at least 3 members of the Workshop Program Committee), 16 papers were selected for presentation and were included in this book.

The 41 contributions published in this book address many topics related to theory and applications of intelligent distributed computing and multi-agent systems, including: agent-based data processing, ambient intelligence, bio-informatics, collaborative systems, cryptography and security, distributed algorithms, grid and cloud computing, information extraction, intelligent robotics, knowledge management, linked data, mobile agents, ontologies, pervasive computing, self-organizing systems, peer-to-peer computing, social networks and trust, and swarm intelligence.

The 5th edition of the Workshop on Multi-Agent Systems Technology and Semantics (MASTS 2013), organized under the framework of the FP7 project “ERRIC: Empowering Romanian Research on Intelligent Information Technologies”, aims to explore novel approaches to multi-agent systems development and the use of multi-agent systems technology in building innovative applications. The goal of the workshop was to present and discuss ideas and developments related to two converging areas of research and applications, namely multi-agent systems and semantic technology.

The 2nd edition of the Workshop on Agents for Clouds (A4C 2013) has been organized under the framework of the FP7 project “mOAIC: Open Source API and platform for multiple clouds”. It aims at promoting exchange and share of experiences, new ideas, and research results about agents programming and technologies applied to Cloud, Grid, and distributed systems.

We would like to thank to Prof. Janusz Kacprzyk, editor of *Studies in Computational Intelligence* series and member of the Steering Committee for their kind support and encouragement in starting and continuing the IDC Symposium Series. We would like to thank to the IDC'2013, MASTS'2013, A4C'2013, and iR'2013 Program Committee members for their work in promoting the event and refereeing submissions and also to all colleagues who submitted papers to IDC'2013, MASTS'2013, A4C'2013, and iR'2013. We deeply appreciate the efforts of two invited speakers Prof. Ngoc Thanh Nguyen and Prof. Michal Pechoucek, and thank them for the interesting lectures. Special thanks also go to organizers of MASTS'2013 (Adina Magda Florea, Amal El Fallah Seghrouchni, John Jules Meyer), A4C'2013 (Salvatore Venticinque), and iR'2013 (Dorian Cojocaru). Finally, we appreciate the efforts of local organizers on behalf of Charles University Prague for organizing and hosting IDC'2013 and the co-located workshops.

Prague, Gyeongsan, Craiova
June 2013

Filip Zavoral
Jason J. Jung
Costin Bădică

Organization

Organizers

Department of Software Engineering, Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic
Software Engineering Department, University of Craiova, Romania

Conference Chair

Filip Zavoral Charles University Prague, Czech Republic

Steering Committee

Janusz Kacprzyk Polish Academy of Sciences, Poland
Costin Bădică University of Craiova, Romania
Frances Brazier Delft University of Technology,

Mohammad Essaaidi The Netherlands
Abdelmalek Essaadi University in Tetuan,
Morocco

Giancarlo Fortino University of Calabria, Italy
Michele Malgeri University of Catania, Italy
Kees Nieuwenhuis Thales Research & Technology,

George A. Papadopoulos
Marcin Paprzycki
The Netherlands
University of Cyprus, Cyprus
Polish Academy of Sciences, Poland

Invited Speakers

Ngoc Thanh Nguyen
Michal Pěchouček
Wroclaw University of Technology, Poland
Czech Technical University, Czech Republic

Program Committee Chairs

Jason J. Jung
Filip Zavoral

Yeungnam University, Korea
Charles University Prague, Czech Republic

Program Committee

Salvador Abreu	Universidade de Evora and CENTRIA
Razvan Andonie	Central Washington University
Costin Bădică	University of Craiova, Software Engineering Department, Romania
Mert Bal	Assistant Professor at the Miami University
Nick Bassiliades	Aristotle University of Thessaloniki
Doina Bein	The Pennsylvania State University
Nik Bessis	University of Derby
Lars Braubach	University of Hamburg
Dumitru Dan Burdescu	University of Craiova
Giacomo Cabri	Università di Modena e Reggio Emilia
David Camacho	Universidad Autonoma de Madrid
Vincenza Carchiolo	Universita di Catania
Phan Cong-Vinh	NTT University
Alfredo Cuzzocrea	ICAR-CNR and University of Calabria
Paul Davidsson	Malmo University
Giuseppe Di Fatta	University of Reading
Luminita Dumitriu	Universitatea Dunarea de Jos din Galati
Barbara Dunin-Keplicz	University of Warsaw
Amal El Fallah Seghrouchni	LIP6 - University of Pierre and Marie Curie
George Eleftherakis	The University of Sheffield International Faculty, CITY College
Vadim Ermolayev	Zaporozhye National Univ.
Mohammad Essaaidi	IEEE Morocco Section
Mostafa Ezziyyani	Faculty of Sciences and Technologies
Adina-Magda Florea	University Politehnica of Bucharest
Giancarlo Fortino	University of Calabria
Stefano Galzarano	University of Calabria (UNICAL), Italy
Maria Ganzha	University of Gdansk, Poland
Marie-Pierre Gleizes	IRIT
Jorge Gomez-Sanz	Universidad Complutense de Madrid
Nathan Griffiths	University of Warwick
Marjan Gusev	Univ. Sts Cyril and Methodius
Michael Hartung	Interdisciplinary Centre for Bioinformatics, University of Leipzig
Jen-Yao Chung	IBM T. J. Watson Research Center
Barna Laszlo Iantovics	Petru Maior University of Tg. Mures

Mirjana Ivanovic	Faculty of Science, Department of Mathematics and Informatics
Jason J. Jung	Yeungnam University
Ioan Jurca	Politehnica University of Timisoara
Igor Kotenko	SPIRAS
Dariusz Krol	Wroclaw University of Technology, Institute of Applied Informatics
Florin Leon	Technical University "Gheorghe Asachi" of Iasi
Antonio Liotta	Eindhoven University of Technology
Alessandro Longheu	DIEEI - University of Catania
Heitor Silverio Lopes	UTFPR
Jose Machado	Universidade Minho
Michele Malgeri	Universita degli Studi di Catania
Giuseppe Mangioni	University of Catania
Yannis Manolopoulos	Aristotle University of Thessaloniki
Viviana Mascardi	CS Department (DISI) - Universit degli Studi di Genova
Amnon Meisels	Ben Gurion University of the Negev
Grzegorz J. Nalepa	AGH University of Science and Technology
Viorel Negru	West University of Timisoara
Peter Noerr	MuseGlobal, Inc.
David Obdržálek	Charles University Prague
Eugenio Oliveira	Faculdade de Engenharia Universidade do Porto - LIACC
Andrea Omicini	Universita di Bologna
Mihaela Oprea	University Petroleum-Gas of Ploiesti, Dept. of Informatics
Carlos Palau	UPV
Marcin Paprzycki	IBS PAN and WSM
Gregor Pavlin	Thales Group
Juan Pavon	Universidad Complutense Madrid
Stefan-Gheorghe Pentiuc	University Stefan cel Mare Suceava
Dana Petcu	West University of Timisoara
Florin Pop	University Politehnica of Bucharest
Radu-Emil Precup	Politehnica University of Timisoara
Shahram Rahimi	Southern Illinois University
Domenico Rosaci	DIMET Department, University Mediterranea of Reggio Calabria
Ioan Salomie	Technical University of Cluj-Napoca
Corrado Santoro	University of Catania - Dipartimento di Matematica e Informatica
Weiming Shen	NRC, Canada
Safeeullah Soomro	BIZTEK Institute of Business & Technology
Giandomenico Spezzano	CNR-ICAR and University of Calabria
Stanimir Stoyanov	University of Plovdiv
Jiao Tao	Oracle

Rainer Unland
 Salvatore Venticinque
 Lucian Vintan
 Martijn Warnier
 Niek Wijngaards
 Jakub Yaghob
 Filip Zavoral

University of Duisburg-Essen, ICB
 Seconda Università di Napoli
 "Lucian Blaga" University of Sibiu
 Delft University of Technology
 D-CIS Lab / Thales Research & Technology
 Charles University Prague
 Charles University Prague

Proceedings Chair

David Bednárek

Charles University Prague, Czech Republic

Organizing Committee

David Bednárek
 Zbyněk Falt
 Sorin Ilie
 Martin Kruliš
 Jakub Yaghob

Charles University Prague, Czech Republic
 Charles University Prague, Czech Republic
 University of Craiova, Romania
 Charles University Prague, Czech Republic
 Charles University Prague, Czech Republic

iR 2013 Workshop Chair

Dorian Cojocaru

University of Craiova, Romania

iR 2013 Workshop Program Committee

Atta Badii
 Theodor Borangiu
 Catalin Buiu
 Gregory Chondrocoukis
 Dorian Cojocaru
 Ioan Doroftei
 Mircea Ivanescu
 Peter Kovacs
 Gheorghe Lazea
 Inocentiu Maniu
 Dan Marghitu
 Gheorghe Mogan
 Florin Moldoveanu
 Sorin Moraru
 Zbigniew Nawrat

School of Systems Engineering University of
 Reading, UK
 Politehnica University of Bucharest, Romania
 Politehnica University of Bucharest, Romania
 University of Piraeus, Greece
 University of Craiova, Romania
 Gheorghe Asachi Technical University of Iasi,
 Romania
 University of Craiova, Romania
 Holografika, Hungary
 Technical University of Cluj, Romania
 Politehnica University of Timisoara, Romania
 Auburn University, USA
 University Transilvania of Brasov, Romania
 University Transilvania of Brasov, Romania
 University Transilvania of Brasov, Romania
 Silesian University of Technology, Gliwice,
 Poland

Doru Panescu	Gheorghe Asachi Technical University of Iasi, Romania
Gheorghe Pentiuc	University "Stefan cel Mare" of Suceava, Romania
Krzysztof Tchon	Wroclaw University of Technology, Poland
Virgil Tiponut	Politehnica University of Timisoara, Romania
Doru Talaba	University Transilvania of Brasov, Romania

A4C Workshop Chairs

Salvatore Venticinque Seconda Università di Napoli

A4C Workshop Program Committee

Alba Amato	Seconda Università di Napoli, Italy
Rocco Aversa	Seconda Università di Napoli, Italy
Pasquale Cantiello	Seconda Università di Napoli, Italy
Giuseppina Cretella	Seconda Università di Napoli, Italy
Massimo Ficco	Seconda Università di Napoli, Italy
Beniamino Di Martino	Seconda Università di Napoli, Italy
Francesco Moscato	Seconda Università di Napoli, Italy
Massimiliano Rak	Seconda Università di Napoli, Italy
Luca Tasquier	Seconda Università di Napoli, Italy
Salvatore Venticinque	Seconda Università di Napoli, Italy

MASTS Workshop Chairs

Adina Magda Florea	University Politehnica of Bucharest, Romania
Amal El Fallah Seghrouchni	Université Pierre & Marie Curie, France
John Jules Meyer	Utrecht University, The Netherlands

MASTS Workshop Program Committee

Costin Bădică	University of Craiova, Romania
Olivier Boissier	ENS des Mines Saint-Etienne, France
Mehdi Dastani	Utrecht University, The Netherlands
Amal El Fallah Seghrouchni	Université Pierre & Marie Curie, France
Adina Magda Florea	University Politehnica of Bucharest, Romania
John Jules Meyer	Utrecht University, The Netherlands
Andrei-Horia Mogos	University Politehnica of Bucharest, Romania

Irina Mocanu
Viorel Negru
Andrei Olaru
Marcin Paprzycki
Stefan Trausan-Matu
Laurent Vercouter
Gerard Vreeswijk
Antoine Zimmermann

University Politehnica of Bucharest, Romania
West University of Timisoara, Romania
University Politehnica of Bucharest, Romania
Polish Academy of Science, Poland
University Politehnica of Bucharest, Romania
INSA Rouen, France
Utrecht University, The Netherlands
ENS des Mines Saint-Etienne, France

Contents

Invited Paper

- Integration Computing and Collective Intelligence** 1
Ngoc Thanh Nguyen

Intelligent Data Processing

- Evolution of a Relational Schema and Its Impact on SQL Queries** 5
Martin Chytil, Marek Polák, Martin Nečaský, Irena Holubová

- Context-Aware Regression from Distributed Sources** 17
Héctor Allende-Cid, Claudio Moraga, Héctor Allende, Raúl Monge

- Incremental Patterns in Text Search** 23
Makoto Yamaguchi, Takao Miura

- REBECCA: A Trust-Based Filtering to Improve Recommendations for B2C e-Commerce** 31
Domenico Rosaci, Giuseppe M.L. Sarné

- Exploration of Document Classification with Linked Data and PageRank** 37
Martin Dostal, Michal Nykl, Karel Ježek

- Matching Users with Groups in Social Networks** 45
Domenico Rosaci, Giuseppe M.L. Sarné

Peer-to-Peer and Cloud Systems

- Semantically Partitioned Peer to Peer Complex Event Processing** 55
Filip Nguyen, Daniel Tovarňák, Tomáš Pitner

A Heuristic to Explore Trust Networks Dynamics	67
<i>Vincenza Carchiolo, Alessandro Longheu, Michele Malgeri, Giuseppe Mangioni</i>	
Resource Scaling Performance for Cache Intensive Algorithms in Windows Azure	77
<i>Marjan Gusev, Sasko Ristov</i>	
Distributed Event-Driven Model for Intelligent Monitoring of Cloud Datacenters	87
<i>Daniel Tovarňák, Filip Nguyen, Tomáš Pitner</i>	
Programming Self-organizing Pervasive Applications with SAPERE ...	93
<i>Franco Zambonelli, Gabriella Castelli, Marco Mamei, Alberto Rosi</i>	
SOMEWHERE2 – A Robust Package for Collaborative Decentralized Consequence-Finding	103
<i>Philippe Chatalic, Andre de Amorim Fonseca</i>	
Distributed Systems and Algorithms	
Heuristic Co-allocation Strategies in Distributed Computing with Non-dedicated Resources	109
<i>Victor Toporkov, Anna Toporkova, Alexey Tselsishchev, Dmitry Yemelyanov</i>	
Distributed Version of Algorithm for Generalized One-Sided Concept Lattices	119
<i>Peter Butka, Jozef Pócs, Jana Pócsová</i>	
Scalable Spatio-temporal Analysis on Distributed Camera Networks ...	131
<i>Kirak Hong, Beate Ottenwälter, Umakishore Ramachandran</i>	
Service-Wide Adaptations in Distributed Embedded Component-Based Systems	141
<i>Luís Nogueira, Jorge Coelho</i>	
Distributed/Parallel Genetic Algorithm for Road Traffic Network Division for Distributed Traffic Simulation	151
<i>Tomas Potuzak</i>	
Environmental Influence in Bio-inspired Game Level Solver Algorithms	157
<i>Antonio Gonzalez-Pardo, David Camacho</i>	
The Impact of the “Nogood Processor” Technique in Scale-Free Networks	163
<i>Ionel Muscalagiu, Horia Emil Popa, Viorel Negru</i>	

Agent-Based Systems

- Knowledge-Based Agent for Efficient Allocation of Distributed Resources** 175
Ebrahim Nageba, Mahmoud Barhamgi, Jocelyne Fayn

- A New Proof System to Verify GDT Agents** 181
Bruno Mermet, Gaele Simon

- Using Trusted Communities to Improve the Speedup of Agents in a Desktop Grid System** 189
Lukas Klejnowski, Sebastian Niemann, Yvonne Bernard, Christian Müller-Schloer

- High-Volume Data Streaming with Agents** 199
Lars Braubach, Kai Jander, Alexander Pokahr

- Strategic Behaviour in Multi-Agent Systems Able to Perform Temporal Reasoning** 211
Matei Popovici, Lorina Negreanu

Intelligent Robots, iR 2013

- Control of a Mobile Robot by Human Gestures** 217
Stefan-Gheorghe Pentiuc, Oana Mihaela Vultur, Andrei Ciupu

- Improving Noise Robustness of Speech Emotion Recognition System** 223
Łukasz Juszakiewicz

- Developing an Avatar Model for Driving Simulators** 233
Razvan-Vlad Vasiliu, Ligia Munteanu, Cornel Brisian

A4C 2013

- Interconnection of Federated Clouds** 243
Massimo Ficco, Luca Tasquier, Beniamino Di Martino

- Effective QoS Monitoring in Large Scale Social Networks** 249
Luigi Coppolino, Salvatore D'Antonio, Luigi Romano, Fotis Aisopos, Konstantinos Tserpes

- Personalized Recommendation of Semantically Annotated Media Contents** 261
Alba Amato, Beniamino Di Martino, Marco Scialdone, Salvatore Venticinque

- Supporting Cloud Governance through Technologies and Standards** 271
Victor Ion Munteanu, Teodor-Florin Fortiș, Adrian Copie

Exploiting Cloud Technologies and Context Information for Recommending Touristic Paths	281
<i>Flora Amato, Antonino Mazzeo, Vincenzo Moscato, Antonio Picariello</i>	
An FPGA-Based Smart Classifier for Decision Support Systems	289
<i>Flora Amato, Mario Barbareschi, Valentina Casola, Antonino Mazzeo</i>	
MASTS 2013	
Agents Modeling under Fairness Assumption in Event-B	301
<i>Irina Mocanu, Lorina Negreanu, Adina Magda Florea</i>	
A Survey of Adaptive Game AI: Considerations for Cloud Deployment	309
<i>Gabriel Iuhasz, Victor Ion Munteanu, Viorel Negru</i>	
From Agent-Oriented Models to Profile Driven Military Training Scenarios	317
<i>Inna Shvartsman, Kuldar Taveter</i>	
An Agent-Based Solution for the Problem of Designing Complex Ambient Intelligence Systems	323
<i>Marius-Tudor Benea</i>	
Agent-Based System for Affective Intelligent Environment	335
<i>Mihaela-Alexandra Puică, Irina Mocanu, Adina-Magda Florea</i>	
An Argumentation Framework for BDI Agents	343
<i>Tudor Berariu</i>	
Using Emotion as Motivation in the Newtonian Emotion System	355
<i>Valentin Lungu, Andra Băltoiu, Șerban Radu</i>	
Distributed Reputation Mechanism Using Semantic Repositories	365
<i>Andreea Urzica, Ileana Bobric</i>	
Author Index	371

Integration Computing and Collective Intelligence

Ngoc Thanh Nguyen

1 Knowledge Integration

Integration is a process in which one of the following aspects should be realised:

- Several objects are merged to give a new element representing them
- Several objects create a union acting as a whole
- Several objects are connected with each other

The first two aspects are most important and most popular [1], [3]. In general, an integration task most often refers to a set of elements (objects) with the same kind of structures, the aim of which is based on determining an element best representing the given. The kinds of structures mean for example relational, hierarchical, table etc. The words "best representation" mentioned above mean the following criteria [4]:

- All data included in the elements to be integrated should be in the result of integration. This criterion guarantees the completeness, that is all information included in the component elements will appear in the integration result.
- All conflicts appearing among elements to be integrated should be solved. It often happens that referring to the same subject different elements contain inconsistent information. Such situation is called a conflict. The integration result should not contain inconsistency, so the conflicts should be solved.
- The kind of structure of the integration result should be the same as of the given elements.

2 Ontology Integration

Integration tasks are very often realised for database integration or knowledge integration. Ontology integration is one of these tasks, which is very often needed to be

Ngoc Thanh Nguyen
Wroclaw University of Technology, Poland
e-mail: Ngoc-Thanh.Nguyen@pwr.edu.pl

done. Ontologies have well-defined structure and it is assumed that the result of ontology integration is also an ontology, therefore usually the first and second criteria are used [5].

It seems that satisfying the first criterion is simple since one can make the sum of all sets of concepts, relations and axioms from component ontologies in the integration process. However, it is not always possible because of the following reasons:

- Appearance of all elements in the integration result may contain inconsistency in the sense that some of the component ontologies may be in conflict and this conflict will be moved to the integration result.
- Summing all elements may cause lose of the ontology structure. Satisfying the second criterion is based on solving conflicts, for example, by using consensus methods. Conflicts between ontologies may be considered referring to the following levels:
 - Conflicts on concept level: The same concept has different fuzzy structures in different ontologies.
 - Conflicts on relation level: The relations for the same concepts are different in different ontologies.

Conflicts mentioned in the this way are very general and inaccurate. We have worked out several methods for ontology conflict resolution regarding these levels.

3 Integration Computing for Building Collective Intelligence

Nowadays it happens very often that for making a decision we rely on collective knowledge, that is knowledge originated from different and autonomous sources, for example, from experts or Internet. Lets consider a situation when someone wants to buy a new car, then he/she will seeks the opinions about the type, model of the car in different forums, social networks, automotive portals, expert systems etc. and on their integrated basis a decision about the purchase will be made. Taking into account the fact that very often the amount of knowledge is very large and the gathered pieces of knowledge are incomplete and inconsistent, one can state that the integration process is a complex task. To make the process effective, a mechanism for knowledge integration is needed to be worked out. For knowledge with complex structures such as hierarchies or graphs the algorithms for integration proposed in the world-wide literature are still not so advanced and need to be developed.

We have proposed a general framework for integration computing referring to determining collective intelligence [4]. We have shown that in general the knowledge of a collective is more proper than the knowledge of its members. This, in turn, proves that a collective is often more intelligent than single units.

An application of integration computing methods in managing data warehouse federations has been worked out and analyzed [2].

References

1. Danilowicz, C., Nguyen, N.T.: Consensus-Based Partitions in the Space of Ordered Partitions. *Pattern Recognition* 21(3), 269–273 (1988)
2. Kern, R., Stolarczyk, T., Nguyen, N.T.: A Formal Framework for Query Decomposition and Knowledge Integration in Data Warehouse Federations. *Expert Systems with Applications* 40(7), 2592–2606 (2013)
3. Nguyen, N.T.: Advanced Methods for Inconsistent Knowledge Management. Springer, London (2008)
4. Nguyen, N.T.: Processing Inconsistency of Knowledge in Determining Knowledge of a Collective. *Cybernetics and Systems* 40(8), 670–688 (2009)
5. Pietranik, M., Nguyen, N.T.: A Method for Ontology Alignment Based Attribute Semantics. *Cybernetics and Systems* 43(4), 319–339 (2012)

Evolution of a Relational Schema and Its Impact on SQL Queries^{*}

Martin Chytil, Marek Polák, Martin Nečaský, and Irena Holubová

Abstract. Since the typical feature of a real-world application is evolution, i.e. changes in user requirements, the problem of change propagation is an important task. In this paper we study its specific part – evolution of a relational database schema and its impact on related SQL queries. The proposed approach shows an ability to model database queries together with a database schema. The feature then provides a solution how to adapt database queries related to the evolved database schema. The proposal was implemented within a complex evolution framework called *DaemonX* and various experiments proving the concept were carried out.

1 Introduction

Since most of the current applications are dynamic, sooner or later the structure of the data needs to be changed and so have to be changed also all related issues. We speak about *evolution* and *adaptability* of applications. One of the aspects of this problem is an adaptation of the respective storage of the data. The adaptation of the storage covers many related issues, such as *database schema evolution* (i.e. retaining system functionality despite schema changes), *database schema integration* (i.e. cases when more database schemas have to be combined together), *data migration* (i.e. a situation when data have to be moved from one system to another), or *adaptation of respective queries* (i.e. reformulation of queries with regard to changes in data schemas).

In this paper we focus on the adaptation of SQL queries. In our case, a change of the underlying database schema can cause that SQL queries over this schema may

Martin Chytil · Marek Polák · Martin Nečaský · Irena Holubová
XML and Web Engineering Research Group, Department of Software Engineering,
Charles University in Prague, Czech Republic
e-mail: chytil.martin@seznam.cz,
{polak,necasky,holubova}@ksi.mff.cuni.cz

* This work was supported by the grant SVV-2013-267312 and GAUK grant no. 1416213.

become inconsistent with the new schema, e.g., a database table has a new name in the new schema, a table column has a new name in the new schema, a database table does not exist in the new schema, a table column does not exist in the new schema, a new table column, which should be used in SQL query, appeared in the new schema, etc. All these presented possibilities lead to *incorrect* SQL queries or queries that do not return the original result. So they should be corrected. For example, suppose an SQL view *PendingOrders*, which returns all information about the pending orders, including all items of the given order. Now, when the name of the column *itemName* is changed (for instance to *productName*), all SQL queries where this column is used have to be checked by a designer and updated respectively.

The proposed approach shows an ability to model database queries together with a database schema model. The feature then provides a solution how to adapt database queries related to the evolved database schema. The proposal was implemented within a complex evolution framework called *DaemonX* and various experiments proving the concept were carried out.

The paper is structured as follows: In Section 2 we describe our database model and in Section 3 we introduce a respective query model. In Section 4 we describe the SQL query visualization model. In Section 5 we introduce the edit operations and respective evolution algorithms. In Section 6 we provide a proof of the concept and we conclude the paper in Section 7.

2 Database Model

The database model we consider is based on the relational database model. Such database model is used as platform-specific according to the *MDA approach* [6] which deals with the idea of separating the specification of the operation of a system from details of the way that system uses the capabilities of its platform. MDA provides an approach for the following actions: Specifying a system independently of the platform that supports, specifying platforms, choosing a particular platform for the system, transforming the system specification into one for a particular platform. Hence, we distinguish a *platform-independent model* (PIM) and *platform-specific model* (PSM).

In this paper, the database model has two roles. Besides the classical PSM for data, it is used as the PIM for a query model described in Section 3. The idea of using the database model this way is simple. From the perspective of an arbitrary query language (not only SQL), the database model creates a basic concept of a database schema. Each query language then creates own platform-specific view of the platform-independent database model.

The PSM database model diagram consists of: *tables*, *columns* which are included in tables and *relationships* visualized as an arrow leading from the referencing table to the referenced table. The full description of the model can be found in [3]. We omit the details and examples of space limitations and general popularity of the model.

3 Query Model

For possibility of evolution of SQL queries related to a given database schema there must exist a mapping between an SQL query and a database schema model. This mapping helps to manage the evolution process to evolve the query related to the evolved database schema. In this chapter we introduce a graph-based SQL query model which is particularly designed for the evolution process. We describe its visualization model, limitations and possibilities. The mapping between the SQL query model and the database schema will be introduced as well. (The algorithm to generate the SQL query from the model can be found in [3].)

The full SQL language syntax was not used in the proposal and for this reason there exist some limitations on the used subset of SQL language:

- Projection operator ‘*’ is banned to use. It is always necessary to enumerate all the columns used in the *SELECT* clause.
- In queries it is possible to use only simple column enumeration, other expressions or functions other than aggregate functions are banned to use. This limitation relates to the *CASE* construct as well.
- The model does not support *UNION*, *INTERSECT* and *EXCEPT* constructs.
- Each condition used in the SQL query is assumed to be in the *conjunctive normal form* (CNF) [5].

The idea of the graph-based model results from papers [8] and [7]. However, in this paper we use a graph-based model for query modeling, in contrast to the mentioned papers, where graph-based model is implemented as a part of the *database management system* (DBMS). The idea of a graph-based model from [8] and [7] is adjusted and extended for the purposes of our approach.

First, each SQL query in the model is represented as a directed graph with particular properties.

Definition 1. (Query Graph). A *query graph* G of the SQL query Q is a directed graph $G_Q = (V, E)$, where V is a set of *query vertices* and E is a set of *query edges*.

Definition 2. (Query Model). A *query model* M of the SQL query Q is a pentad $M_Q = (G_Q, T_V, T_E, \tau_V, \tau_E)$, where G_Q is a query graph $G_Q = (V, E)$, T_V is a set of vertex types {*AggregateFunction*, *Alias*, *BooleanOperator*, *CombiningSource*, *ComparingOperator*, *ConstantOperand*, *DataSource*, *DataSourceItem*, *From*, *FromItem*, *GroupBy*, *Having*, *OrderBy*, *OrderByType*, *QueryOperator*, *Select*, *SelectItem*, *Where*}, T_E is a set of edge types {*Alias*, *Condition*, *ConditionOperand*, *DataSource*, *DataSourceAlias*, *DataSourceItem*, *DataSourceParent*, *FromItem*, *FromItemParent*, *FromItemSource*, *GroupBy*, *GroupByColumn*, *Having*, *MapColumn*, *MapSource*, *OrderBy*, *OrderByColumn*, *SelectColumn*, *SelectQuery*, *SourceTree*, *Where*}, a function $\tau_V : V \rightarrow T_V$ assigns a type to each vertex of the query graph G_Q and a function $\tau_E : E \rightarrow T_E$ assigns a type to each edge of the query graph G_Q .

A *query vertex* represents a particular part of the SQL query, e.g. a database table, a table column, a comparing operator in condition, a selected column in the *SELECT*

clause, etc. A *query edge* connects parts of the SQL query together and gives a particular semantics to this connection. For instance the edge connecting a *From* vertex and a *Where* vertex means that the query contains a *WHERE* clause represented by the *Where* vertex.

Each query graph can be logically divided into smaller subgraphs. These subgraphs are called *essential components*. Each essential component has a visual equivalent in the query visualization model (described in Section 4). There exist the following essential components: *DataSource*, *From*, *Select*, *Condition*, *GroupBy*, *OrderBy*. For instance, the simplest SQL queries of the form ‘*SELECT projection FROM table*’ require only *DataSource*, *From* and *Select* components. Fig. 1 illustrates a simple example of the modeled *GROUP BY* clause. The example is equivalent to the following parts of the SQL query:

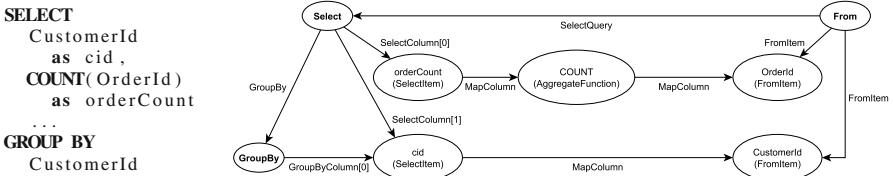


Fig. 1 An example of a simple model of the *GROUP BY* clause

(All of these components contain subcomponents. The full list of the subcomponents with their descriptions can be found in [3].)

4 SQL Query Visualization Model

Although the graph-based query model can describe any SQL query, it is relatively complex. Even the query model for a simple SQL query contains a lot of vertices and edges (see Fig. 1). For this reason we proposed a visualization model, which simplifies an underlying query model for users and model creation.

4.1 Visualization Model Components

The visualization model is divided into so-called *essential visual components*. Each essential component described in Section 3 has its visual equivalent by some essential visual component, so each visual component represents a part of the SQL query graph model. We distinguish the following visual components: *DataSource*, *QueryComponent* and *Component Connection*.

DataSource Visual Component. A *DataSource* visual component visualizes a *DataSource* essential component. Each *DataSource* has a name, which clearly identifies the given data source. The content of the *DataSource* component is a list

of *DataSourceItem* visual components. The *DataSource* visual component itself corresponds to the *DataSource* vertex. The *DataSourceItem* visual components correspond to the *DataSourceItem* vertices. For example, the rectangle *Customer* in Fig. 2 shows an example of *DataSource* visual component. The example represents a database table *Customer* with columns: *customerId*, *firstname*, *lastname*, *email* and *phone*.

QueryComponent Visual Component. A *QueryComponent* is a universal visual essential component, which represents parts of the SQL query. A basic appearance of all query components looks the same. We distinguish the following types of query components according to the clause of the SQL query they represent: *Select*, *From*, *Where*, *GroupBy*, *Having*, *OrderBy*. For example, the rectangle *Where* in Fig. 2 illustrates an example of the *QueryComponent* visual component. The example shows visualization of the *WHERE* clause.

Component Connection. A *Component Connection* does not correspond directly to any essential component of the query graph. Instead, it covers a connection of two essential components to finish the correct and complete query graph. We distinguish the following types of connections: *DataSource* → *From*, *Select* → *From*, *Where* → *From*, *GroupBy* → *Select*, *Having* → *From*, *OrderBy* → *From*. For example, Fig. 2 shows a visualization model of a more complex SQL query. The modelled SQL query is the following one:

```

SELECT
    c.firstname , c.lastname ,
    a.street , a.city ,
    a.postcode
FROM
    Customer as c JOIN
    Address as a
    ON c.customerId = a.customerId
WHERE
    (c.firstname = 'John' OR
     c.firstname = 'Jane')
    AND
    (c.lastname = 'Doe')
ORDER BY
    c.customerId ASC,
    c.lastname ASC, a.postcode DESC
  
```

For comparison, the underlaying query graph model consists of 45 vertices connected by 87 edges.

4.2 Mapping to the Database Model

Since the database model consists of tables and its columns which we can interpret as a general source of data, we have a direct mapping from the database model to the query model. We do not consider database relationships between database tables in the database model. For the purpose of the query model they are not important.

The mapping between the database model and the query model is described as follows:

- **Database table → *DataSource*.** The database table in the database model is mapped to the *DataSource* visual component which corresponds to the *DataSource* vertex of the underlaying query graph.

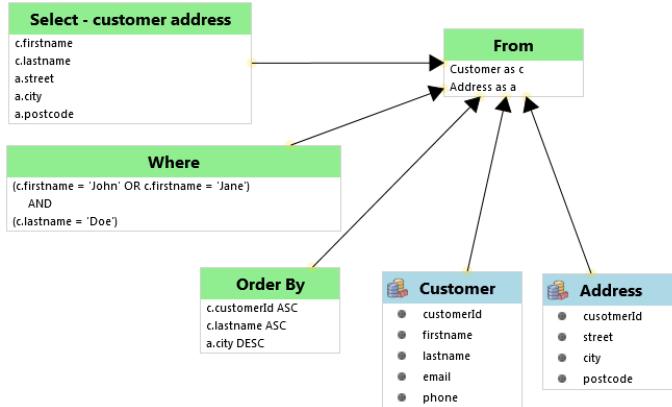


Fig. 2 An example of a visual model of a more complex SQL query

- **Table column → *DataSourceItem*.** The table column in the database model is mapped to the *DataSourceItem* visual component which corresponds to the *DataSourceItem* vertex of the underlaying query graph.

Note that the mapping does not preserve keys (primary keys, foreign keys) and any other column attributes like *NOT NULL* or *UNIQUE*. For the purpose of the data querying this property is insignificant.

4.3 Mapping of Operations

In this section we describe the mapping of the operations of database and query models used in the evolution process.

Definition 3. (Operation). An *operation* is a general function $f : (M, C) \rightarrow M'$, where M is a particular model intended to change, C represents a context describing a change of the model M and M' is a modified model.

Definition 4. (Atomic Operation). An *atomic operation* is the minimal indivisible operation. It can be used to create *composite operations*.

All changes in the database model are done via atomic operations. All atomic operations in the database model which have an impact on the SQL query model are translated by the evolution process into corresponding atomic operations in the SQL query model.

4.3.1 Database Model Operations

Let us suppose a database model M_D , which is a set of tables T_i , $i \in [1, n]$. Each table $T_i \in M_D$ has a name T_{iN} and a set T_{iC} , which is a set of columns c_j , $j \in [1, n_i]$. Each column c_j has a name c_{jN} .

- **Renaming Database Table** ($\alpha_T : (T_i, m) \rightarrow T'_i$): The operation returns table T'_i where $T'_{i_N} = m$ and $T'_{i_C} = T_{i_C}$.
- **Removing Database Table** ($\beta_T : (M_D, T_i) \rightarrow M'_D$): The operation removes database table $T_i \in M_D$ from the database model M_D . It returns database model M'_D where $M'_D = M_D \setminus \{T_i\}$.
- **Creating Table Column** ($\gamma_C : (T_i, c_j) \rightarrow T'_i$): The operation adds the column c_j into table T_i . It returns table T'_i where $T'_{i_N} = T_{i_N}$ and $T'_{i_C} = T_{i_C} \cup \{c_j\}$.
- **Renaming Table Column** ($\alpha_C : (c_j, m) \rightarrow c'_j$): The operation returns column c'_j where $c'_{j_N} = m$.
- **Removing Table Column** ($\beta_C : (T_i, c_j) \rightarrow T'_i$): The operation removes column $c_j \in T_i$ from the table T_i . It returns table T'_i where $T'_{i_N} = T_{i_N}$ and $T'_{i_C} = T_{i_C} \setminus \{c_j\}$.

4.3.2 SQL Query Model Operations

Let us suppose a query model M_Q , whose query graph G_Q consists of a set of *Data-Sources* $D_i, i \in [1, k]$ and other components, which are not important for our purpose. Each *DataSource* $D_i \in M_Q$ has a name D_{i_N} and a set D_{i_l} , which is a set of *DataSourceItems* $d_j, j \in [1, k_i]$. Each *DataSourceItem* d_j has a name d_{j_N} .

- **Renaming DataSource** ($\alpha_D : (D_i, m) \rightarrow D'_i$): The operation returns *DataSource* D'_i where $D'_{i_N} = m$ and $D'_{i_l} = D_{i_l}$.
- **Removing DataSource** ($\beta_D : (M_Q, D_i) \rightarrow M'_Q$): The operation removes *DataSource* $D_i \in M_Q$ from the query model M_Q . It returns query model M'_Q where $M'_Q = M_Q \setminus \{D_i\}$.
- **Creating DataSourceItem** ($\gamma_I : (D_i, d_j) \rightarrow D'_i$): The operation adds *DataSourceItem* d_j into *DataSource* D_i . It returns the *DataSource* D'_i where $D'_{i_N} = D_{i_N}$ and $D'_{i_l} = D_{i_l} \cup \{d_j\}$.
- **Renaming DataSourceItem** ($\alpha_I : (d_j, m) \rightarrow d'_j$): The operation returns *DataSourceItem* d'_j where $d'_{j_N} = m$.
- **Removing DataSourceItem** ($\beta_I : (D_i, d_j) \rightarrow D'_i$): The operation removes *DataSourceItem* $d_j \in D_i$ from the *DataSource* D_i . It returns *DataSource* D'_i where $D'_{i_N} = D_{i_N}$ and $D'_{i_l} = D_{i_l} \setminus \{d_j\}$.

4.3.3 Complex Operations

More complex operations like *Split*, *Merge*, *Move* done in the database model can be propagated to the SQL query model as well. In the SQL query model these operations are simply composed of the mentioned atomic operations.

5 Change Propagation in the Graph

The database model operations described in Section 4.3.1 have an impact on the queries in the SQL query model. During the evolution process, changes in the database model have to be propagated to the SQL query model, where the modeled queries are adapted to the current database model. However, sometimes a

simple direct propagation is not correct. For instance, if a new column is added to the database table, we may not want to add a new column to the *SELECT* clause of the query. For this reason we propose so-called *propagation policies*, which influence the behavior of the propagation. The policies are defined for the vertices of the query graph, which participate in the change distribution process.

We distinguish the following propagation policies:

- **Propagate:** This policy allows to perform the change directly. Subsequently, the propagation is passed on the following vertices in the change process.
- **Block:** This policy does not allow to perform the change. The subsequent propagation is stopped and the following vertices in the change process are not visited.
- **Prompt:** The system asks a user which of the two above policies should be used to continue.

5.1 Query Graph Operations

As mentioned before, the SQL query model operations described in Section 4.3.2 are atomic operations, i.e. they cannot be divided into smaller operations. In fact, these atomic operations consist of many smaller steps called *graph operations*, which modify the query graph of the SQL query model. In the following definitions G_Q represents a query graph $G_Q = (V, E)$. We distinguish the following graph operations:

- **CreateVertex** ($\gamma_v : (G_Q, v) \rightarrow G'_Q$): The operation returns graph $G'_Q = (V \cup \{v\}, E)$.
- **CreateEdge** ($\gamma_e : (G_Q, v_{source}, v_{target}, e_{type}) \rightarrow G'_Q$): The operation creates edge $e = (v_{source}, v_{target})$ such that $EdgeType(e) = e_{type}$ and returns graph $G'_Q = (V, E \cup \{e\})$.
- **RemoveVertex** ($\beta_v : (G_Q, v) \rightarrow G'_Q$): The operation returns graph $G'_Q = (V \setminus \{v\}, E \cap \binom{V \setminus \{v\}}{2})$.
- **RemoveEdge** ($\beta_e : (G_Q, e) \rightarrow G'_Q$): The operation returns graph $G'_Q = (V, E \setminus \{e\})$.
- **ChangeLabel** ($\lambda : (v, l) \rightarrow v'$): The operation returns query vertex v' , where vertex type $v'_{type} = v_{type}$ and label $v'_L = l$. (For instance, it returns the *DataSourceItem* vertex with a new name.)
- **ChangeConnectionType** ($\eta : (C, t) \rightarrow C'$): This operation returns *Combine-Source* vertex C' with connection type $C'_T = t$.
- **ResetContent** ($\rho(G_Q, C)$): Since the visualization model visualizes the query graph, they have to be synchronized. This operation is used to signal the parent visual component C that a change in the query graph G_Q has been done and the content of the visual component has to be updated.

These atomic functions are combined in the set of so-called *composite operations* which are called in the evolution process. All these functions with theirs algorithms are described in [3].

An Example of Creating a Table Column. First, Algorithm 1 creates a new *FromItem* vertex in the corresponding *From* vertex and connects it with appropriate vertices. Subsequently it traverses to the *Select* vertex, where it creates corresponding vertices and edges using algorithm *DistributeCreatingDatasourceItemSelect* (see [3]). Finally, it traverses to the *OrderBy* vertex, where it creates corresponding vertices and edges using algorithm *DistributeCreatingDatasourceItemOrderBy*. Fig. 3 depicts adding of a new *DataSourceItem OrderDate* to the *DataSource* component using Algorithm *DistributeCreatingDatasourceItem* and to the *From* component using Algorithm 1. In Fig. 3 the original elements are black and the new elements of the query graph are highlighted with a red color.

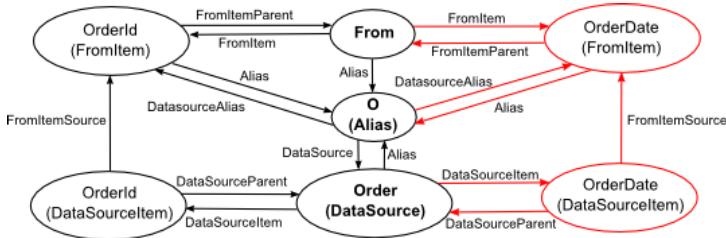


Fig. 3 An example of adding new *DataSourceItem* to the *DataSource* and to the *From* components

Algorithm 1. DistributeCreatingDatasourceItemOnAlias

Require: Alias vertex A, change context C
Ensure: Graph operations to create new *DataSourceItem*.

- 1: $fromVertex \leftarrow A.GetNeighbour(DataSourceAlias)$
- 2: $newItem \leftarrow \text{new FromItem}(C.Name)$
- 3: $C.Plan \leftarrow CreateVertex(C.G_Q, newItem)$
- 4: $C.Plan \leftarrow CreateEdge(C.G_Q, C.Originator, newItem, FromItemSource)$
- 5: $C.Plan \leftarrow CreateEdge(C.G_Q, newItem, A, Alias)$
- 6: $C.Plan \leftarrow CreateEdge(C.G_Q, fromVertex, newItem, FromItem)$
- 7: $C.Plan \leftarrow CreateEdge(C.G_Q, newItem, fromVertex, FromItemParent)$
- 8: $C.Originator \leftarrow newItem$
- 9: $selectVertex \leftarrow fromVertex.GetNeighbour(SelectQuery)$
- 10: **if** $selectVertex \neq \text{null}$ **then**
- 11: *DistributeCreatingDatasourceItemOnSelect(selectVertex, C)*
- 12: **end if**
- 13: $orderByVertex \leftarrow fromVertex.GetNeighbour(OrderBy)$
- 14: **if** $orderByVertex \neq \text{null}$ **then**
- 15: *DistributeCreatingDatasourceItemOnOrderBy(orderByVertex, C)*
- 16: **end if**
- 17: $C.Plan \leftarrow ResetContent(C.G_Q, fromVertex)$

Algorithm *DistributeCreatingDatasourceItemSelect* creates a new *SelectItem* vertex in the *SELECT* clause. Then it checks, whether the *GroupBy* vertex exists. If it does, it connects the *GroupBy* vertex with the new *SelectItem* vertex. Finally, it traverses to all Alias vertices of dependant queries and applies already mentioned Algorithm 1.

6 Proof of the Concept

The full experimental implementation of the presented approach was incorporated into the *DaemonX* framework [4]. (In fact, Figs. 2, 4 and 5 are screen shots of the application.) Our approach adds two new plug-ins into the framework. The first plug-in is a plug-in for SQL query modeling described in Section 4 (a screen-shot from this plug-in is depicted in Fig. 2). The second plug-in is used for evolution propagation from the PSM database model to the SQL query model (as described in Section 5).

Since there are no existing applications which provide similar abilities, to be able to compare our approach we used an existing database project *Adventure Works* [2]. We modeled in the database model a set of tables of a given database schema (the list of them can be found in [3]). From this database model we derived to the query model tables as *DataSources*, which we used to model queries and views. Next, we applied various operations over the database model to simulate propagation to the query model. After the propagation the new queries were inspected if they correspond to the expected results.

Due to space limitations we will use some simplification. We present an example of adding new column to the table and its propagation to the related *GroupBy* query. Suppose the following SQL query which model is depicted in Fig. 4:

```
SELECT
    d.GroupName ,
    COUNT(d.Name)
        as NumberOfDepartments
FROM
    HumanResources . Department as d
GROUP BY
    d.GroupName
HAVING
    COUNT(d.Name) > 2
ORDER BY NumberOfDepartments DESC
```

A new column *GroupID* was added to the table *HumanResources.Department*. This change was propagated into the complex *GroupBy* query. The new column was added to all its components. The original query was transformed by algorithm *DistributeCreatingDatasourceItem* (described in [3]) to the new query (its model is depicted in Fig. 5):

```
SELECT
    d.GroupName ,
    COUNT(d.Name)
        as NumberOfDepartments ,
    d.GroupID
FROM
    HumanResources . Department as d
GROUP BY
    d.GroupName , d.GroupID
HAVING COUNT(d.Name) > 2
ORDER BY
    NumberOfDepartments DESC,
    d.GroupID ASC
```

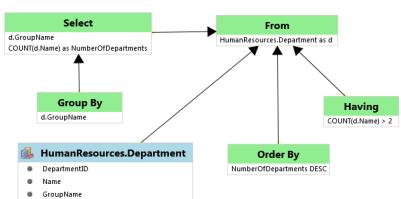


Fig. 4 The model of the complex usage of complex *GroupBy* query

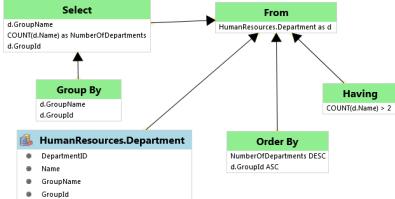


Fig. 5 The updated model of the complex *GroupBy* query

7 Conclusion

In this paper we have focussed on a specific part of the problem of change management of applications – evolution of a relational schema and its propagation to SQL queries. The main contribution of our approach is the ability to model SQL queries concurrently with the respective schema, to analyze changes performed in the database model and to update the queries to preserve their compatibility and correctness. Changes in the database schema model are propagated immediately to the SQL query using mutual mapping.

Even though the approach is complex and robust, there exists a number of open problems. A natural first extension is towards more complex constructs we have omitted. Next, we can consider also other query languages, such as XQuery [9] for XML data or SPARQL [1] for RDF data. And, last but not least, there is an important aspect of semantics of the data and queries which may highly influence the propagation mechanism.

References

1. SPARQL Query Language for RDF. W3C (2008), <http://www.w3.org/TR/rdf-sparql-query/>
2. Adventure Works team. Adventure Works 2008R2 (November 2010), <http://msftdbprodsamples.codeplex.com>
3. Chytil, M.: Adaptation of Relational Database Schema. Master's thesis, Charles University in Prague (2012), <http://www.ksi.mff.cuni.cz/~holubova/dp/Chytil.pdf>
4. DaemonX-Team. DaemonX (June 2001), <http://daemonx.codeplex.com>
5. Korovin, K.: CNF and Clausal Form. In: Logic in Computer Science, Lecture Notes (2006)
6. Miller, J., Mukerji, J.: MDA Guide Version 1.0.1 (2003), http://www.omg.org/mda/mda_files/MDA_Guide_Version1-0.pdf
7. Papastefanatos, G., Vassiliadis, P., Simitsis, A., Aggistalis, K., Pechlivani, F., Vassiliou, Y.: Language Extensions for the Automation of Database Schema Evolution. In: Cordeiro, J., Filipe, J. (eds.) ICEIS (1), pp. 74–81 (2008)
8. Papastefanatos, G., Vassiliadis, P., Vassiliou, Y.: Adaptive Query Formulation to Handle Database Evolution. In: Proceedings of the CAiSE Forum (2006)
9. W3 Consortium. XQuery 1.0: An XML Query Language, W3C Working Draft (November 12, 2003), <http://www.w3.org/TR/xquery/>

Context-Aware Regression from Distributed Sources

Héctor Allende-Cid, Claudio Moraga, Héctor Allende, and Raúl Monge

Abstract. In this paper we present a distributed regression framework to model data with different contexts. Different context is defined as the change of the underlying laws of probability in the distributed sources. Most state of the art methods do not take into account the different context and assume that the data comes from the same statistical distribution. We propose an aggregation scheme for models that are in the same neighborhood in terms of statistical divergence. We conduct experiments with synthetic data sets to validate our proposal. Our proposed algorithm outperforms other models that follow a traditional approach.

Keywords: Distributed Regression, Context-aware Regression, Divergence Measures.

1 Introduction

The field of Distributed Data Mining (DDM) has been very active and is enjoying a growing amount of attention since it was first proposed. Most of the current DDM techniques treat the distributed data sets as a single virtual table and assume that there is a global model which could be generated if the data were combined or centralized, completely neglecting the different semantic contexts that this distributed data sets may have [1]. If we see this as a statistical learning problem, we deal with samples of data that follow different underlying laws of probability. Loosely speaking Machine Learning models try to find a function which relates a given output

Héctor Allende-Cid · Héctor Allende · Raúl Monge

Departamento de Informática, Universidad Técnica Federico Santa María

Avenida España 1680, Valparaíso, Chile

e-mail: vector@inf.utfsm.cl

Claudio Moraga

European Centre for Soft Computing, 33600, Mieres, Asturias, Spain

Claudio Moraga

TU Dortmund University, 44220 Dortmund, Germany

y with an input vector x . The classic Machine Learning approach is related with the estimation of the joint probability distribution $H(\underline{X}, \underline{Y})$. The joint probability distribution can be decomposed in the conditional probability distribution and the marginal one ($H(\underline{X}, \underline{Y}) = F(\underline{Y}|\underline{X})G(\underline{X})$). In this paper, context is defined as the joint probability distribution that governs each data source. The main task addressed in this proposal will be the one of Distributed Regression (DR). Approaches to deal with problems of distributed regression are harder to find in the DDM literature than problems of distributed classification. Even harder to find are algorithms that deal with the problem when the data sources have different underlying laws of probability. This proposal could help a virtual organization, e.g. a temporary network of companies/entities that come together quickly to exploit fast changing opportunities, with each partner contributing what it is best at. The next section will present a brief view of the state of the art related to this work. In section 3 we present the proposed algorithm which will be able to address the problem presented above. In section 4 we show some experimental results. The last section is devoted to discuss the results and to make some conclusions.

2 State of the Art

There is a large number of works done in the last decade in the field of Distributed Data Mining. A large fraction of DDM algorithms focuses on combining predictive models. This approach has emerged from empirical experimentation due to a requirement for higher prediction accuracy. Recently, several researchers treat distributed learning systems as a centralized ensemble-based method [1]. Several learning algorithms are applied at each local site, using separate training data to mine local knowledge. A new data point is then classified/predicted from the predictions of all local sites using ensemble methods such as stacking, boosting, majority voting, simple average, or winner-takes-all methods. In general, DDM approaches apply ensemble methods to minimize the communication costs and to increase the performance of the system predictions.

Yan Xing et al. [1] have proposed a series of algorithms based on, what the authors call, a meta-learning approach to deal with the regression problem addressing the context heterogeneity case. The authors propose a meta-learning-based hierarchical model that is able to be successfully used in distributed scenarios with context heterogeneity. The definition of context in this work is the variance that the distributed sources have in their outputs, thus neglecting the context change in the input space. The authors claim that this change of context between distributed sites is random. For a more complete review on the state of the art of Distributed Data Mining Algorithms, please refer to [1].

Like it was explained in the previous section the differences in context are the differences among probability distributions. For this we are going to use the well-known (h, ϕ) -divergence measures [1]. With this divergence measure we are going to perform a hypothesis test, where the null hypothesis checks whether the probability

distributions of 2 samples have the same set of parameters versus the alternative hypothesis that they have different set of parameters. (Mainly, mean and co-variance)

For further details please refer to [1].

3 Proposal

In this section we present the proposed algorithm. It can be broken down into the following phases:

3.1 Learning

1. *Phase 1 - Local Learning.* Suppose that there are k distributed data sets. At each node N_i , where $i = 1, \dots, k$, we use an available learning algorithm to train a local predictive model L_i from data source D_i of that node. This proposal is a meta-model that could be used for classification or regression problems. The choice of the learning algorithm is not restricted to any particular kind. Also we get the mean vector and the variance-covariance matrix at each node. The local training data consists in an n dimensional input vector $((x_{i1}, x_{i2}, \dots, x_{in}))$ and a response variable y_i .
2. *Phase 2 - Model and information transmission.* Each node N_i , where $i = 1, \dots, k$ receives the model parameters, mean vector, variance-covariance matrix and number of samples of the other nodes. A hypothesis test is performed based on the (h, ϕ) -divergence for each pair of local data sets D of the nodes N_i and N_j , where $i \neq j$. These hypotheses tests will check if the data of the current local node follow the same underlying law of probability of the rest of the nodes. In this proposal we focus only in the marginal statistical distribution of the input space. The result is a binary vector (h_i) with k binary variables, which indicates the nodes which follow the same underlying law of probability of D_i . In other terms we will refer to this vector as the Neighborhood vector based on divergence measures. Also compute the Hamming weight m_i for each node N_i . E.g. If there are 5 distributed nodes D_1, D_2, D_3, D_4 and D_5 , and we are checking if node D_1 has the same distribution as the rest, and only the variables $h_1(1), h_1(2)$ and $h_1(4)$ are equal to 1, meaning that the data contained in the nodes N_2 and N_4 follow the same distribution as in node N_1 , then $m_1 = 3$.
3. *Phase 3 - Generation of second-stage learners.* Since every node N_i has a copy of the other local models, each local model L_l , $l = 1, \dots, k$, contained in node N_i is trained with the local data from that node (D_l). Each of the local models in N_i outputs a response variable \hat{y}_{ij} with the local data D_i , where $j = 1, \dots, k$. Each local node applies then a stacked learning algorithm G_i (second-stage model) which is trained with the outputs of all the local models $(\hat{y}_{i1}, \hat{y}_{i2}, \dots, \hat{y}_{ik})$ and the real response variable y_i , obtained from the training data of the node D_i . This is inspired in the stacking model of Wolpert [1].

3.2 Predicting

1. *Final output of the proposed model.* The output of our model is the following: Whenever a new example arrives at a node N_i , we compute all the outputs of the local models that are stored in this node. We have an apriori information of which of the other nodes have data following a close underlying law of probability of the current node, which is reflected in the binary vector mentioned above (h_r , where $r = 1, \dots, k$). Then the output of the local models in this node are transmitted to only the other nodes which have a non-zero label in this vector. The final output of the model is the sum of all the G_i model outputs that received the output of all the local models in the current node divided by the Hamming weight m_r . E.g. in the example presented in Fig. 1 the final output of the model is the mean of models G_1 , G_3 and G_k , because only the variables $h_2(1)$, $h_3(3)$ and $h_3(k)$ are distinct from zero. The Hamming weight m_r in this case is equal to 3.

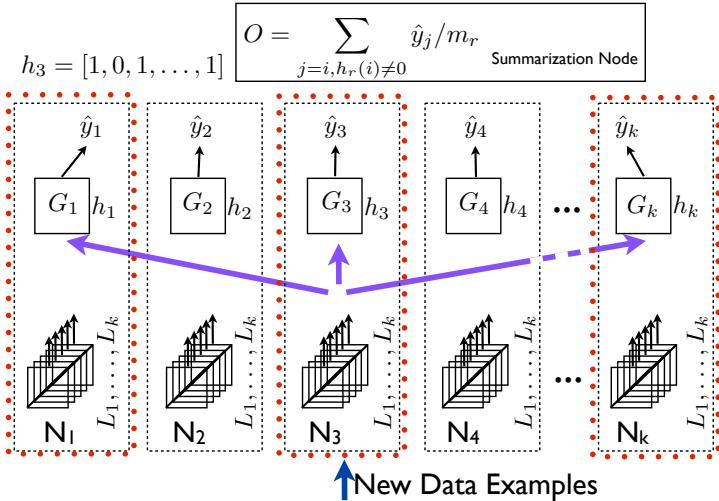


Fig. 1 Architecture of the proposed system

4 Experimentation

In this section we present the results obtained by our proposal with two synthetic experiments. The reason why we did not test our proposal with real data sets is because real data sets which have the characteristics that we are dealing in this proposal are hard to find. But since we are trying to prove that our proposed algorithm is able to detect different contexts, synthetic data sets are useful for this purpose. The data sources created for the first synthetic experiment were generated from a nonlinear function with an input dimension of 2 (Equation (1)). The second

experiment consisted in data coming from a nonlinear function with an input dimension of 3 (Equation (2)).

$$y = \frac{\sin(x_1) \sin(x_2)}{x_1 x_2} \quad (1)$$

$$y = (1 + x^{0.5} + y^{-1} + z^{-1.5})^2 \quad (2)$$

For the first experiment we tested 2 scenarios: Data coming from multivariate normal distributions with same variance-covariance matrix Σ but different mean vector μ , and different variance-covariance matrix Σ but the same mean vector μ . For the second one we generated datasets with both differences, different mean vectors and different variance-covariance matrices. The number of distributed sources were 5, 10 and 20. The distributed sources were generated with 2 different sets of parameters. The number of sources from set of parameters (μ_1, Σ_1) and (μ_2, Σ_2) was generated randomly. The number of examples of each source was set to 1000. The maximum likelihood estimator was used to estimate the parameters of the gaussian distributions. As a local model a Feedforward neural network with one hidden layer, and 3, 5, 10 neurons were tested. We chose the number of neurons which gave us the best results in terms of mean squared error (MSE) in 10 experimental runs. For a detailed description of the experimental configuration please refer to [1].

The divergence measures used in this proposal are the Kullback-Leibler and Renyi divergence measures. The α -value used to perform the hypothesis test to establish the neighborhood for each of the distributed sources was 0.95.

To compare our proposal we tested 3 different algorithms:

- Local G_i models. Stacked models trained with the output of all local models with the data of node N_i . (*Single*)
- Pseudo-Ensemble of all G_i models. (*Ensemble*)
- Aggregation of G_i models within the neighborhood of N_i . (*Proposal*)

As can be seen in Table 1 our proposal outperforms in every case, the other models. Column *Exp* shows 4 different experimental configurations (Difference between the sets of parameters was incremented in each configuration). Both divergences measures built the same neighborhood, so the results were the same for both. From the results obtained in this first synthetic experiment we can see that it is very important to take into account the statistical distributions that govern each data set to perform a better estimation. If we do not take this into account the performance using all the G_i models generated in each source can even worsen the results we could obtain only using the single G_i models. The error reported in Table 1 are the mean of 10 experimental runs. The performance error used is the mean MSE of all distributed data sets. The results of the second synthetic experiment show similar results as the one obtained with the first synthetic experiment. For the results and further details of the second experiment see [1].

Table 1 Mean MSE of 10 experimental runs. Data sources with probability distributions with different mean vectors (left) and data sources with probability distributions with different variance-covariance matrices (right).

Exp	NS	Single	Ensemble	Proposal	Single	Ensemble	Proposal
1	5	7,85e-04	7,82e-04	7,68e-04	5,6703e-04	5,617e-04	5,1278e-04
	10	3,88e-04	3,72e-04	3,64e-04	3,1548e-04	3,0457e-04	2,8615e-04
	20	0,0013	4,65e-04	4,30e-04	4,1741e-04	4,3338e-04	4,1529e-04
2	5	0,0015	0,0019	0,0013	0,0055	0,0084	0,0050
	10	0,0014	0,0016	0,0013	0,0061	0,0137	0,0051
	20	0,0031	0,0033	0,0029	0,0050	0,011	0,0043
3	5	0,0320	0,0532	0,0314	0,0103	0,0552	0,0097
	10	0,0099	0,0281	0,0094	0,0113	0,0933	0,106
	20	0,0149	0,0661	0,0143	0,0082	0,0551	0,0070
4	5	0,0197	0,0627	0,0154	0,0041	0,1024	0,0040
	10	0,0114	0,0621	0,0105	0,0091	0,1465	0,0040
	20	0,0122	0,0643	0,0109	0,0067	0,1067	0,0062

5 Conclusions

In this proposal we present a distributed regression approach that is able to detect different contexts in the input space, thus improving the performance of local models in the task of regression from distributed sources. As the results show, it is very important, not to neglect the different contexts that are present in the distributed sources. Using classic ensemble-based approaches including all sites, that are available in the distributed data mining literature, we obtain worse results, than the ones we obtain by using only local models in appropriate Neighborhoods. In future approaches we will additionally take advantage of the differences in the conditional statistical distribution of the output space, thus improving even more the results obtained here. Finally we will test our proposal with real data sets.

Acknowledgments. This work was supported by the following research grants: Fondecyt 1110854 and DGIP-UTFSM. The work of C. Moraga was partially supported by the Foundation for the Advancement of Soft Computing, Mieres, Spain and by the CICYT Spain, under project TIN 2011-29827-C02-01.

Reference

1. Allende-Cid, H., Moraga, C., Allende, H., Monge, R.: Regression from distributed sources with different underlying laws of probability. Technical Report, European Centre for Soft Computing, Mieres, Asturias, Spain (available upon request, 2013)

Incremental Patterns in Text Search

Makoto Yamaguchi and Takao Miura

Abstract. In this investigation, we propose a new kind of text retrieval, called *dynamic KMP*, where the pattern becomes longer incrementally. We introduce several auxiliary information for efficient management and examine how well the approach the works.

Keywords: Dynamic KMP, KMP algorithm, incremental patterns, text retrieval.

1 Motivation

In current internet world, there have been much amount of information in a variety of text strings such Blog and twitter messages. To extract useful information correctly, timely and efficiently among them, it is not enough to utilize traditional techniques of database nor information retrieval. In information retrieval, one of the fundamental functions comes from *pattern* matching where, given query patterns, we examine data in our database and explore answers. To specify query intents, we may introduce regular expression or approximate string matching as query patterns by using special algorithms (such as KMP and BM methods) or sophisticated data organization (such as suffix trees). In the latter case, suffix tree/array provides us with compact data structures by analyzing information in advance so that we may obtain efficient retrieval, but all the suffixes are stored into the tree/array. In text retrieval, we often change query patterns dynamically depending upon the explosion of information. For example, whenever we obtain useful information from huge amount of log information about stock market, we like to change our query pattern immediately to obtain relevant information in more specific manner and we come to suitable information soon. We may also navigate relevant DNA which are

Makoto Yamaguchi · Takao Miura
HOSEI University, Dept.of Elect.& Elect. Engr.
3-7-2 KajinoCho, Koganei, Tokyo, 184–8584 Japan
e-mail: makoto.yamaguchi.9y@stu.hosei.ac.jp,
miurat@hosei.ac.jp

similar to previous query results. In these cases, we need new techniques for text retrieval which provide us with incremental query patterns. Here in this work, we extend KMP algorithm to change incremental patterns dynamically while keeping same query costs.

This work is organized as follows. We review several algorithms for text retrieval in section 2 while we examine some properties of Maximum Prefix (ML) in section 3. In section 4 we discuss how to manage incremental change of query patterns based on KMP algorithm. We examine some experiments in section 5 to see how well our model works. We conclude our work in section 6.

2 Text Matching

There have been many kinds of efficient algorithms proposed for pattern matching from huge amount of text information (called *text matching* or *text retrieval*) where we give text string as a pattern to be examined. Among others, we have two well-known approaches, KMP method and BM method[1].

Knuth-Morris-Pratt (KMP) algorithm is an efficient algorithm for pattern matching on text string. The basic idea comes from a fact that, by analyzing patterns in advance, we may obtain a new position after matching-failure and skip intermediate characters so that we recover some overhead for the analysis.

Let us show our running example in a table 1. Given a pattern “*issis*”, we examine a text information “mississippi” to obtain positions of matching. Whenever we get mismatching (marked by “*”), we skip several characters according to sliding information obtained from the pattern. For example, we have a mismatching character “*p*” at line 4, and we skip 3 characters of the patterns to the right and continue the process at that “*p*”. KMP method allows us to examine text information only once.

Table 1 Examining “*issis*”

	Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	Text	mississippi																
1)	i*																	
2)	issis																	
3)	iss*																	
4)	issis*																	
5)	is*																	
6)	i*																	
7)	issis*																	
8)	is*																	
9)	i*																	
10)	i*																	
11)	i*																	

Table 2 MP and MPL of “*ississii*”

	<i>m</i>	MPL (<i>m</i>)	MP (<i>m</i>)
“i”	0	-1	-
“is”	1	-1	-
“iss”	2	-1	-
“issi”	3	0	“i”
“issis”	4	1	“is”
“ississ”	5	2	“iss”
“ississi”	6	3	“issi”
“ississii”	7	0	“i”

Boyer-Moore (BM) algorithm also requires analysis of patterns in advance but differs from KMP where patterns are examined in a backward manner so that we may skip more. Compared to KMP algorithm, sometimes we get more much efficient but not stable results.

In suffix trees, on the other hand, it takes $O(N^2)$ space to construct. Suffix array improves this deficiency but needs $O(N \log N)$ or $O(N^2 \log N)$ for the construction with complicated algorithms[4]

3 Maximum Prefix for KMP Algorithm

Let us describe how KMP algorithm does work using Maximum Prefix (MP). Given a pattern $\text{PATTERN}[0..m+1]$ of length $m + 2$ which consists of elements in $V = \{c_1, \dots, c_V\}$, we examine text information for matching and assume the first mismatch at $\text{PATTERN}[m+1]$. KMP method provides us with how many characters we *skip* to re-start matching, because all the $\text{PATTERN}[0..m]$ are exactly equal to the text information. We analyze the pattern to see the longest prefix of the pattern, called *Maximum Prefix*(MP), that matches the suffix. We define *Maximum Prefix* (MP) of a pattern $\text{PATTERN}[0..m]$ ($m > 0$):

$$\begin{aligned} \text{MPL}(m) &= \max \{ k \mid \text{PATTERN}[0..k] = \text{PATTERN}[(m-k)..m], 0 \leq k < m \} \\ \text{MPL}(m) &= -1, \text{ if there exists no such prefix.} \\ \text{MP}(m) &= \text{PATTERN}[0..k], k = \text{MPL}(m) \text{ if } \text{MPL}(m) \geq 0, \text{ null otherwise.} \end{aligned}$$

Assume $\text{MPL}(0) = -1$. Note that we have MP of length $k + 1$ if $k = \text{MPL}(m)$ where $\text{MP}(m) = \text{PATTERN}[0..m]$ if $\text{MPL}(m) \geq 0$.

Let us illustrate MP and MPL of a pattern $\text{PATTERN} = \text{"ississippi"}$ in table 2. We have $\text{MPL}(0) = -1$ by definition. We also have $\text{MPL}(1) = -1$, $\text{MPL}(2)=-1$ since there is no prefix equal to the corresponding suffix in "is", "iss". In a case of "issi" (length 4), we see the longest match of $\text{PATTERN}[0..0] = \text{i}$ and $\text{PATTERN}[3-0..3] = \text{i}$ so that we have $\text{MP}(3) = \text{i}$ and $\text{MPL}(3) = 0$. Similarly in "issis", "ississ" and "ississi", we have $\text{MPL}(4) = 1$, $\text{MP}(4) = \text{is}$, $\text{MPL}(5) = 2$, $\text{MP}(5) = \text{iss}$ and $\text{MPL}(6) = 3$, $\text{MP}(6) = \text{issi}$ respectively. However, a case of "ississii" (length 8) shows the longest match of $\text{PATTERN}[0..0] = \text{i}$ and $\text{PATTERN}[7-0..7] = \text{i}$ so that we have $\text{MP}(7) = \text{i}$ and $\text{MPL}(7) = 0$.

After getting mismatch of PATTERN at position 0 to text information at some position w , we start matching again at $w + 1$ with the full pattern of PATTERN. When we find mismatching of PATTERN at position $m + 1 > 0$ to text information at some position w , we utilize $\text{MP}(m) = \text{PATTERN}[0..m]$. In fact, if $\text{MPL}(m) \geq 0$, we know the prefix $\text{MP}(m)$ is equal to the suffix (to the position m) so we can skip this part, and we continue matching at $\text{MPL}(m) + 1$ of PATTERN to the position w of the text information. If $\text{MPL}(m) < 0$, no matching happens and we start matching again from the beginning of PATTERN to the position w of the text information.

We examine $D = \text{"mississippi"}$ and $\text{PATTERN} = \text{"issis"}$ in a table 1. At position 6 of D we see mismatch of $\text{PATTERN}[2] = \text{s}$ (see line 3). Since we have "is" matched with $D[4..5]$ but $\text{MPL}(1) = -1$, we start again with the beginning of PATTERN. As shown in line 4, we have $\text{PATTERN}[4] = \text{s}$ mismatched with $D[10]$. Because $\text{MPL}(3) = 0$, we can skip 3 (=3-0) characters.

In line 5, we have mismatching of $\text{PATTERN}[1] = "s"$ at position 10 of D . Since $\text{MPL}(0) = -1$, we can skip one character.

When we build MP of length $k+1$ where $k = \text{MPL}(m)$, we assume the prefix has never overlapped the suffix, then we must have $(k+1) \leq (m+1)/2$. If $k=0, m>0$, it holds $\text{PATTERN}[(m-i)..m] \neq \text{PATTERN}[0..i]$ for $i=1,..,m$. For example, as in a table 2, given $\text{PATTERN}[0..4] = "issis"$ ($m=4$), we have $\text{MP} = "is"$, $\text{MPL}(4) = k = 1$ and $(k+1) \leq (m+1)/2$ because no overlap happens.

Conversely, if $(k+1) > (m+1)/2$, we must have the prefix and the suffix with some overlap. Let $\alpha\beta$ be MP of length $k+1$ where β means the overlap. Assume α is longer than β , then β appears in the beginning of α , that is, β is the prefix of α . Assume otherwise. Then α appears in β many times and $\beta = \alpha \cdots \alpha\alpha'$ where α appears at least one time and α' is the prefix of α . Thus we must have $\beta = \alpha \cdots \alpha\alpha'$ if we have if $(k+1) > (m+1)/2$ (the overlap happens). In a table 2, we have $k=\text{MPL}(6)=3$ with $\text{PATTERN}[0..6] = "ississi"$. In this case, the longest match prefix $\text{PATTERN}[0..3] = "issi"$ has the overlap $\text{PATTERN}[3..3] = "i"$ with the longest match suffix $\text{PATTERN}[3..6] = "issi"$. Then MP $\text{PATTERN}[0..3]$ consists of $\alpha\beta$ where $\alpha = \text{PATTERN}[0..2] = "iss"$ and $\beta = \alpha "i"$. Note $\beta ("i")$ appears as a prefix part in $\alpha ("iss")$.

4 Constructing Patterns Incrementally

In KMP algorithm, we utilize MPL. That is, given $0 \leq m < M$ and $\text{PATERN}[0..M]$, In this section, we show $\text{MPL}(m+1) \leq \text{MPL}(m)+1$. At the same time we show how to construct $\text{MPL}(m+1)$ (and $\text{MP}(m)$) using $\text{MPL}(0), \dots, \text{MPL}(m)$. By definition, we have $\text{MPL}(0) = -1$ and let us note the values $\text{MPL}(m)$ increase by 1 or decrease in a table 2.

Let $X = \text{PATTRN}[m+1], k = \text{MPL}(m)$ and $\alpha = \text{PATTERN}[0..k]$ if $k \geq 0$ and $\alpha = ""$ if $k = -1$. Also let $Y = \text{PATTRN}[k+1]$, i.e., a character at $k+1$.

Assume $X = Y$. Then the suffix α X is equal to $\alpha Y = \text{PATTERN}[0..(k+1)]$ and we have $\text{MPL}(m+1) \geq \text{MPL}(m)+1$. If $k' = \text{MPL}(m+1) > \text{MPL}(m)+1 = k+1$, there exists a prefix $\beta = \text{MP}(m+1)$ such that $\beta = \text{PATTERN}[0..k']$, $k' = \text{MPL}(m+1)$, $\text{PATTERN}[k'] = X$ and $\beta = \text{PATTERN}[(m+1-k')..(m+1)]$. Since $X = \text{PATTRN}[m+1]$, $\text{PATTERN}[0..(k'-1)] = \text{PATTERN}[(m+1-k')..m]$ is a prefix of length k' in $\text{PATTERN}[0..m]$. Let us note $k' > k$ and this contradicts the definition of $k = \text{MPL}(m)$ at the position m .

For example, we construct $\text{MPL}(m+1)$ where $m=4$ and $\text{PATTERN}[0..4] = "issis"$. Let us add $X = "s"$ into the pattern at the position 5. Note we have $\text{MP}(4) = "is"$ and $\text{MPL}(4) = 1$ in a table 2. In the new pattern $\text{PATTERN}[0..5] = "ississ"$, we see $Y = \text{PATTERN}[\text{MPL}(4)+1] = \text{PATTERN}[2] = "s"$ and thus $\text{MPL}(5) = \text{MPL}(4)+1=2, \text{MP}(5) = "iss"$.

Assume $X \neq Y$. Given a query pattern $\alpha = \text{PATTERN}[0..(m+1)]$, we must have $\text{MPL}(m+1) \leq \text{MPL}(m)$. In fact, $k = \text{MPL}(m)$ and assume $k \geq 0$. Since $X \neq Y$, the suffix $\text{PATTERN}[(m+1-k)..(m+1)]$ can't be equal to the prefix

PATTERN [0.. k] of α . This means this prefix can't be PATTERN [0.. $MPL(m+1)$] and we have $MPL(m+1) < MPL(m)+1 (= k+1)$. Assume $k = MPL(m) = -1$. Then $MPL(m+1) = MPL(m) = -1$.

For instance, let $m = 6$ and PATTERN [0..6] = "ississi". To insert $X = "i"$ into the pattern, we like $MPL(7)$. By definition and a table 2, we have $MPL(6)=3$ and the longest match prefix at $m = 6$, MP(6), is PATTERN [0.. $MPL(6)$] = "issi". The character $X = "i"$ doesn't appear in the pattern at $MPL(6)+1$, in fact, we have PATTERN [$MPL(6)+1$] = PATTERN [4] = "s". The longest match prefix at $m = 7$, MP(7), is "i" and we have $MPL(7)=0$.

Now let us describe how to construct $MPL(m+1)$ incrementally using PATTERN [0.. M] and $MPL(0), \dots, MPL(m)$. The approach is called *dynamic KMP* algorithm. Here we assume a large memory for patterns, PATTERN [0.. M], $M >> m \geq 0$, and we insert X at PATTERN [$m+1$]. To construct $MPL(m+1)$, it is possible to give $MPL(m+1) = MPL(m)+1$ if $X = PATTERN[MPL(m)+1]$. Assume $X \neq PATTERN[MPL(m)+1]$. When $MPL(m) = -1$, we must have $MPL(m+1) = -1$ by definition. So we assume $MPL(m) \geq 0$ in the following. We like the longest match prefix of PATTERN [0.. $MPL(m)$] but KMP method doesn't seem suitable because we explore the prefixes of PATTERN [0.. m] shorter than $MPL(m)+1$ but we can't take advantages of $MPL(0), \dots, MPL(m-1)$. In dynamic KMP method, we introduce two kinds of auxiliary information, skip1 and skip2 over a set of characters $V = \{c_1, \dots, c_V\}$ to reduce complexity of the processes, where skip1 is an array of size $M+1$ and skip2 of size V . We define $skip1(i)$ as a position prior i where a character PATTERN [i] appears prior i , and also define $skip2(i)$ as a position where c_i appears finally in PATTERN [0.. m]:

$skip2(i) = j, j = \max_j \{ j \mid c_i = PATTERN[j] \}$, and -1 if there is no c_i in the pattern.

$skip1(i) = j, j = \max_j \{ j \mid PATTERN[i] = PATTERN[j], j < i \}$ and -1 if there is no such j in the pattern.

Let us illustrate how the algorithm goes with skip1/skip2 of the patterns of "ississi" and "ississii" in a table 3. When we add "i" to "ississi", we give $skip2["i"]$ to $skip1[7]$ and 7 to $skip2["i"]$.

Table 3 Constructing skip1/skip2

		0 1 2 3 4 5 6 7			0 1 2 3 4 5 6 7
PATTERN ($m=6$) ississi			PATTERN ($m=7$) ississii		
skip1	-1 1 0 2 4 3		skip1	-1 1 0 2 4 3 6	
skip2			skip2		
"i"	6		"i"	7	
"s"	5		"s"	5	

Initially we give -1 to all of $\text{skip1}, \text{skip2}$. Given a new pattern $\text{PATTERN}[0..(m+1)]$ with a new character at $m+1$, we examine all the prefixes of $\text{PATTERN}[0..k]$ using skip1 , skip2 where $k = \text{MPL}(m)$ to construct $\text{MPL}(m+1)$ as follows:

- (1) If $X = \text{PATTRN}[m+1]$, put $j = \text{skip2}(X)$.
- (2) Repeat $j = \text{skip1}(j)$ until $j < k$ while $j \neq -1$.

If $j = -1$, no prefix is found and return -1 . Assume otherwise.

- (3) If $\text{PATTERN}[0..j] = \text{PATTERN}[(m+1-j)..(m+1)]$, the answer is j . Otherwise put $j = \text{skip1}(j)$ and go back to the process 3.

Let us discuss the time complexity issues at *each incremental pattern*. It is possible to obtain $\text{skip1}, \text{skip2}$ incrementally on $O(1)$, It takes $O(1)$ for the process 1 in the above algorithm and $O(m)$ for the process 2. In the process 4, it takes $O(j)$ to examine $\text{PATTERN}[0..j]$ to each repetition, and $O(\text{MPL}^2(m))$ in total. Practically MPL is small very often and few problem arises. It is possible to work efficiently by using KMP/BM methods, but we need complicated process for the preparation and it seems better to do that in a naive manner.

5 Experimental Results

5.1 Preliminaries

In this experiment, we assume incremental pattern and evaluate how well query goes and how about the pattern construction and the management. When the pattern grows (to the length $m+1$), we construct $\text{MPL}(m+1)$ for $\text{PATTERN}[0..(m+1)]$.

We examine *Reuter-21578* corpus[3] because the text documents have been well-formed, well-known and well-analyzed for a variety of evaluation purpose. The corpus consists of 22 data files in SGML format including 21578 news articles over 1.3 million words distributed in 1987. Each article has been tagged and classified into 90 topics by hand. Every article has size of 1KB or 4KB. We have selected 2568 articles randomly from the corpus, tag removed and combined into one of 3MB string. We have also selected 4 articles from them and put them together with duplicates into one as query patterns of size 160,092 bytes, Especially we take an article of 3090 bytes and put 5 times into the beginning of the query pattern for the purpose of longer MPLs.

As shown in a table 4, the prefix of 12366 bytes is equal to the part of +3093, which means some overlap. In fact, the first 3092 bytes are duplicated 4 times after the article in the query. The situation is marked with "A" in the table. Similarly we put "B" (once) and "C" (twice) in the table. There are 26 times duplicates of the article, as well as other 3 articles appear many times. Figure 1 outlines how Maximum Prefix goes.

In this experiment, we assume incremental pattern and evaluate how well query goes and how about the pattern construction and the management. When the pattern grows (to the length $m+1$), we construct $\text{MPL}(m+1)$ for $\text{PATTERN}[0..(m+1)]$.

Table 4 Maximum Prefix of Pattern

Offset	Length	
3093	12366	A
19546	3090	B
29282	3096	B
36469	3096	B
43562	6186	C
53927	6180	C
62658	12360	A
80120	3090	B
89856	3096	B
97041	3090	B
102682	6180	C
128800	6180	C
137531	3096	B
144716	3090	B
157003	3090	B

Table 5 MPL construction

Offset	Dynamic	Additional	Static
10000	9999	0	49995000
20000	10213	252	150029219
30000	10421	4	250131360
40000	10414	126	350228016
50000	10209	126	450316458
60000	10000	0	550394594
70000	10418	8	650441214
80000	10422	16	750473038
90000	10421	4	850556194
100000	10207	63	950653122
110000	10834	12	1050713526
120000	10010	0	1150802866
130000	10005	0	1250925191
140000	10418	8	1350991722
150000	10623	67	1451072620
160000	10005	0	1551162227

Whenever the pattern grows, we may have two kinds of approaches to adjust the problem, *static* construction and *dynamic* construction of MPL. By static approach, we mean we construct $\text{MPL}(0), \dots, \text{MPL}(m+1)$ from scratch to each step. Once we find mismatch at j in $\text{PATTERN}[0..m+1]$, we skip checking the first $\text{MPL}(j)$ characters since we know $\text{PATTERN}[0..j-1]$ appear in the text. To obtain MPL, we examine all the possibilities (i.e., $j = 0, \dots, m+1$) to see the maximum length where the prefix and suffix are identical¹. Since the suffix has been changed, the previous values of MPL are not applicable now. Clearly it takes $O(m)$ for each j , and $O(m^2)$ in total. On the other hand, by dynamic approach, we apply dynamic KMP technique to the construction of MPL. Especially we construct MPL incrementally but not from scratch. Note that there can be no difference of query efficiency between the two approaches because we apply same KMP algorithm for text query, but the difference comes from only MPL construction.

5.2 Results and Discussion

Let us illustrate the results in table 5 which show how many characters are examined for MPL construction to every 10000 characters of the query word (160092 bytes), one for primary step and another for additional step through `skip1`. We also show the static case for the comparison. The situation is outlined in a figure 2. We see, in dynamic approach, it keeps constant values about 10000 while static

¹ We like the maximum value k where a pattern string from k to $j-1$ is equal to the top $j-k$ characters. We put -1 if no matching found.

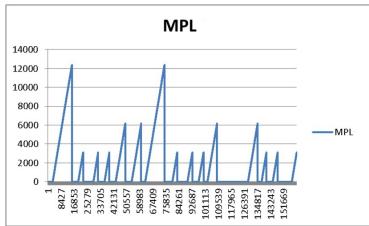


Fig. 1 MPL distribution (0...,160092)

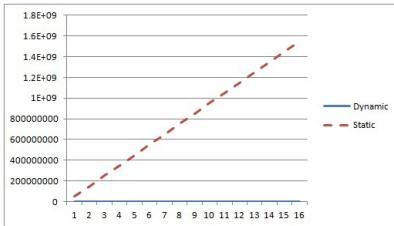


Fig. 2 The number of Comparison

approach increases linearly. This means the numbers of references are $O(M)$ and $O(M^2)$ respectively for the length M of query pattern.

Figure 1 contains many *short* matching. During MPL grows, we examine few long pattern as in table 5, and we examine characters incrementally. Since we have $-1, 0, 1$ and 2 in MPL table except the article of 3090 bytes, we can obtain fast examination for MPL construction once we get mismatch situation.

News corpus may be expected to contain many quotation but few happens practically. Also it seldom happens to go through skip links many times. In fact, “Additional” comparison in table 5 arises at most 252 times to 10000 characters and we have few overhead. That is the reason of $O(M)$ in practice for dynamic MPL construction.

6 Conclusion

In this investigation, we have proposed dynamic KMP algorithm for incremental text search. Our experimental results say that it takes linear time to the length of query pattern. Compared to naive (static) approach, we get dramatic improvement in practice. Though we can improve additional step of MPL construction based on BM method, the approach may cause complicated processing to each pattern increment and it is not suitable for our case.

References

1. Ishihata, K.: Algorithm and Data Structure. Iwanami Publishing (1989) (in Japanese)
2. Manber, U., Myers, G.: Suffix arrays: a new method for on-line string searches. SIAM Journal on Computing 22(5), 935–948 (1993)
3. Reuter Corpus,
<http://www.daviddlewis.com/resources/testcollections/reuters21578>
4. Ukkonen, E.: On-line construction of suffix trees. Algorithmica 14(3), 249–260 (1995)
5. Weiner, P.: Linear pattern matching algorithm. In: 14th Annual IEEE Symposium on Switching and Automata Theory, pp. 1–11 (1973)

REBECCA: A Trust-Based Filtering to Improve Recommendations for B2C e-Commerce

Domenico Rosaci and Giuseppe M.L. Sarné

Abstract. Recommender systems usually support B2C e-Commerce activities without to provide e-buyers with information about the reputation of both products and interlocutors. To provide B2C traders with suggestions taking into account gossips, in this paper we present REBECCA, a fully decentralized trust-based B2C recommender system that also guarantees scalability and privacy. Some experiments show the advantages introduced by REBECCA in generating more effective suggestions.

1 Introduction

Business-to-Consumer (B2C) activities play a relevant role in the e-Commerce (EC) [2] and recommender systems (RS) [21] can support both customers and merchants with suggestions about products and services. To perform such a task, a representation (*profile*) of customer's interests and preferences needs and it can be exploited to implement RSs by using a multi-agent system (MAS) [19] to suitably consider users' orientations [3, 10, 14]. However, MASs need to consider their "social" interactions for allowing an agent to maximize (minimize) its income (outcome) by selecting its interlocutors [7] also by using a trust-based approach [17, 18].

In this paper, we present **R**Eputation-**B**ased **E**-Commerce **C**ommunity of **A**gents (**REBECCA**), to support the generic merchant M in generating suggestions for the generic customer C , based on a collaboration between their associated agents. More in detail, (i) a merchant agent works as a hybrid RS [21] while (ii) a customer agent filters its suggestions based on the customers' satisfaction (i.e. "reputation") about products provided by other agents, weighted in terms of reliability and reputation in the MAS. Many RSs, as [1, 9], consider reliability (reputation) to generate suggestions to present only relevant information according to the personal point of view of other trusted users To this aim, several trust models consider user's orientations

Domenico Rosaci · Giuseppe M.L. Sarné

University Mediterranea of Reggio Calabria, 89123 Reggio Calabria, Italy

e-mail: {domenico.rosaci, sarne}@unirc.it

[5, 8, 16] in computing reliability and/or reputation combining them in a single trust measure [4, 6, 13, 15]. In [11] it is obtained by using a real parameter $\beta \in [0, 1]$, where 1 (0) means all the relevance to reliability (reputation), while REBECCA also considers both the satisfaction of C about a given product and those of the customers for presenting to C only suggestions for high-reputed products. The decentralized architecture makes REBECCA scalable with respect to the MAS size and preserves the C 's privacy since (i) each merchant agent monitors the C 's behaviour in visiting its site and (ii) each customer agent locally computes its trust measures.

Finally, the results of some experiments performed by using a JADE prototype of REBECCA clearly show its advantages, in terms of suggestions effectiveness.

2 The REBECCA Framework

In REBECCA each customer C and each merchant M is associated with a personal agent (resp., c and m) using a common *Catalogue K* storing each *product category* (pc) and its *product instances* (pi). Agents monitors C 's site visits to each pi and pc to build their profiles and measure the associated interests. To determine collaborative filtering suggestions, each agent m computes the similarity among its visitors by using the interest values shown in each offered pi . Finally, c stores an internal trust model of the products that is based both on the direct past experiences and on suggestions coming from each other customers. More in detail, each user U (either customer or merchant) is associated with an agent a that manages a *User Profile* (UP) consisting of the tuple $\langle WD, BD, TD \rangle$, where:

- WD contains the system parameters α , β and γ (see below), and the current K .
- BD describes the U 's past behaviour. Thus for C (M), BD stores, for each visited (offered) pc : (i) the interest Cr in that pc and (ii) L_{pc} , a list of elements l_{pi} associated with a $pi \in pc$, visited by C (offered by M). Each l_{pi} stores some data about the C 's behaviour (the behaviour of all the customers visiting the site) in visiting pi and, in particular, Pr represents the interest in pi .
- TD is only in the C 's profile and consists of 4 arrays of real values set in $[0, 1]$, namely: (i) SAT stores the C 's global satisfaction for each pi based on the other customers' evaluations; (ii) $RECC$ contains the suggestions of other customer agents provided to c about the expertise of other customers agents on a pc ; (iii) REP stores customer agents reputation rates computed using suggestions; (iv) REL contains the reliability values computed by c using its direct experiences;

Note that the meaning of Cr (Pr) for C is his interest rate in the linked pc (pi) but for M it is that of all the visitors of his site.

2.1 Computing Customer Interests and Product Reliability

When C visits the M 's site then the C 's (M 's) agent c (m) monitors the C 's behaviour to computes the interest rates Pr (Cr) for each visited $pi \in pc$ (pc) based on both the time spent in visiting the Web page containing pi (pc) and the time distance

between two consecutive visits. As a consequence, $Pr = \alpha \cdot \delta \cdot Pr + (1 - \alpha) \cdot t$ (resp. $Cr = \alpha \cdot \delta \cdot Cr + (1 - \alpha) \cdot t$), where the past C 's interest for the involved pi (pr) is weighted by α and δ (i.e., $\delta = (365 - \Delta)/365$ and decreases based on the number of days (Δ) between two consecutive visits).

To evaluate the trust value of each pi which C is interested in, the C 's agent requires some recommendations to a subset AS of agents in its MAS, considered as experts in that pc which pi belongs to. For each agent $d \in AS$, each recommendation about d , related to the category pc and coming from the generic agent e is stored in $RECC(e, d, pc)$. Basing on these recommendations, c computes the reputation of d in pc , as $REP(d, pc) = \sum_{e \in AS} RECC(e, d, pc) / |AS|$.

Moreover, the agent c computes the reliability degree of a customer agent d for a pc , considering each opinion $op(d, pi)$ provided by d about each pi belonging to pc , and comparing such opinion with the *feedback* $feed(pi)$ that c provides about pi . Formally:

$$REL(d, pc) = \beta \cdot REL(d, pc) + (1 - \beta) \cdot \frac{1}{q_e \cdot |AS|} \cdot \sum_{e \in AS} \sum_{j=1}^{q_e} \frac{|op(d, pi_j) - feed(pi_j)|}{op(d, pi_j)} \quad (1)$$

where β is a real value, belonging to $[0, 1]$, that represents the importance that c gives to the past evaluations of the reliability with respect to the current evaluation.

Finally, the agent c can compute the product satisfaction of $pi \in pc$ coming from the other customer agents weighted by means of their reliability and reputation integrated in a unique trust measure using a coefficient γ to weight the importance of the reliability vs the reputation. Formally:

$$SAT(pi) = \frac{\sum_{d \in AS} (op(d, pi) \cdot (\gamma \cdot REL(d, pc) + (1 - \gamma) \cdot REP(d, pc)))}{|AS|} \quad (2)$$

3 The REBECCA Recommendation Algorithm

The recommendation algorithm of REBECCA (Figure 1) runs on merchants and customers sides and exploits a *content-based* (CB) and *collaborative filtering* (CF) approaches, using the trust measures described in the previous section.

More in detail, when a customer C visits a merchant site M , the function *Recommend*, executed by the merchant agent m , receives the M 's profile and returns the lists CB and CF of pi that m will recommend to c . To this purpose, m calls the function *extract_pc* receiving the *BD* section of the M 's profile for returning the list $L1$ (with all the $pc \in K$ offered by M) that is sent to c , which answer with the list $L2$ including the v (an integer parameter set by C) pc contained in $L1$ that better meet the C 's interests, ordered by Cr values. When $L2$ is received by m , *contentbased_pi* is called by M and returns a list CB containing, for each of the

$v pc \in L2$, at most y (y is a parameter set by M) pi having the highest rates. To generate CF suggestions the array of lists PC is built by m ; each element is a list associated with a customer i , that visited the site, and stores the v most interesting pc . To this aim, m calls the function `customersInterests`, receiving in input the BD data of the m profile and v . Each array element $PC[i]$ is a list built by (i) using the Cr values, (ii) ordering them in a decreasing order and (iii) selecting, for each user, the $v pc$ with the highest Cr values. Then the function `collaborativefiltering_pi` is called by m ; it receives the list $L2$ (provided by c), the array of lists PC , the BD section of the m profile and the two integers z and x (set by M). For selecting the z agents most similar to c , this function exploits PC to compute the similarity degree between c and each customer that interacted with M in the past. For each of the z most similar customers, at most $x pi$ having the highest Pr rates are inserted in the list CF . The lists CB and CF are returned to m and sent to c .

The function `categoriesOfInterest` is executed by c when it receives from m the list $L1$ and calls `extract_pc` to obtain the list PCI with each pc of interest for C . Then, `intersection_pc` computes the intersection $L1 \cap PCI$ that is stored in $PCI*$. Finally, `select_pc` receives the list $PCI*$ and v for returning the first v product categories having the highest Cr values that are returned into the list $L2$.

Finally, `TBFiltering` is executed on the c side, accepting as input the lists CB and CF sent by m that computed them by exploiting `Recommend`. This function joins CB and CF in the list P and by means of `select_pi` provides to prune such lists from each pi considered as not interesting based on its $SAT(pi)$ value (see Section 2.1). Then `select_pi` returns each $pi \in P$ having a $SAT(pi) > \eta$ (a parameter set by C). This product instances are inserted in a list FP returned to c .

```

void Recommend(merchantAgent m, ListOfProductInstancesCB, ListOfProductInstancesCF) {
    ListOfProductCategories L1=extract_pc(m.UP.BD);
    void send(ListOfProductCategories L1, c.Address);
    void receive(ListOfProductCategories L2, int v, int y);
    ListOfProductInstances CB=contentbased_pi(ListOfProductCategories L2, m.UP.BD, int y);
    ListsOfCustomersInterests PC[] = customersInterests(m.UP.BD, int v);
    ListOfProductInstances CF=collaborativefiltering_pi(ListOfProductCategories L2,
                                                       ListsOfCustomersInterestsPC[], m.UP.BD, int z, int x); }

ListOfProductCategories categoriesOfInterest(ListofProductCategories L1) {
    ListOfProductCategories PCI=extract_pc(c.UP.BD);
    ListOfProductCategories PCI*=intersection_pc(L1, PCI);
    ListOfProductCategories L2=select_pc(PCI*,v);
    return L2; }

ListOfProductInstances TBFiltering(ListofProductInstances CB,CF, real η) {
    ListOfProductInstances P=union_pi(CB,CF);
    ListOfProductInstances FP=select_pi(P, η);
    return FP; }

```

Fig. 1 The REBECCA recommendation algorithm

4 Experiments and Conclusions

This section presents some experiments devoted to show the REBECCA effectiveness in generating useful suggestions to support customers and merchants in their B2C activities. These experiments have been realized by using a JADE (jade.tlab.com) prototype involving 23 XML EC Web sites, a common *Catalogue* (described by a XML Schema) 852 product instances belonging to 10 product categories and a set of 48 real users monitored in their REBECCA B2C activities. To obtain an initial profile of the customers' orientation, the first 10 sites has been used without suggestion support and trusted customer agents. Profiles and trust information are used by the merchant agents to generate suggestions for the other 10 sites. Experiments involved REBECCA, the trust-based recommender system presented in [9] (identified as RS-TB), and the recommender systems EC-XAMAS [3] and TRES [12] that do not consider trust information.

To measure the effectiveness of the proposed suggestions, we have inserted in a list, called *A*, each *pi* suggested by the merchant agent and in a list, called *B*, the corresponding customer's choices, and we have compared the corresponding elements in the two lists. We adopted the standard performance metrics precision and recall [20] and set the customer and merchant parameters v, y, z and x to 2, 3, 2 and 3, respectively.

The results (see Table 1) show that REBECCA in terms of precision (resp. recall) performs 19 (resp. 15) percent better than EC-XAMAS, 9 (resp. 10) percent better than TRES and 20 (resp. 19) percent better than RS-TB. In conclusion, experiments show as REBECCA leads to generate more effective recommendations with respect the other systems mainly due to the use of trust information. Indeed, deactivating the use of the trust-based filtering, represented by the function `TBFILTERING`. The result thus obtained, represented in Table 1, show that the performances of REBECCA significantly decrease.

As conclusion, in an EC community can be present unreliable or misbehaving interlocutors. To this aim, this paper has been presented REBECCA, a trust-based recommender system acting as CB and CF recommender to support B2C traders by considering customers' orientations, products reputation and traders' trustworthiness. The distributed architecture makes efficient the computation of trust measures and suggestions, gives high scalability and preserves customers' privacy. Some interesting experimental results have shown the advantages of REBECCA.

Table 1 Performances of REBECCA (with (A) and without (B) trust measures), RS-TB, EC-XAMAS and TRES

	REBECCA (A)	REBECCA (B)	RS – TB	EC-XAMAS	TRES
Pre	0.892	0.783	0.819	0.698	0.802
Rec	0,878	0.795	0.807	0.721	0.778

Acknowledgement. This work has been partially supported by the TENACE PRIN Project (n. 20103 P34XC) funded by the Italian Ministry of Education, University and Research.

References

1. Avesani, P., Massa, P., Tiella, R.: Moleskiing. It: a trust-aware recommender system for ski mountaineering. *Int. J. for Infonomics*, 20 (2005)
2. Chesbrough, H.: Business Model Innovation: Opportunities and Barriers. *Long Range Planning* 43(2-3), 354–363 (2010)
3. De Meo, P., Rosaci, D., Sarnè, G.M.L., Ursino, D., Terracina, G.: EC-XAMAS: Supporting e-Commerce Activities by an XML-Based Adaptive Multi-Agent System. *Applied Artificial Intelligence* 21(6), 529–562 (2007)
4. Gómez, M., Carbó, J., Benac-Earle, C.: An Anticipatory Trust Model for Open Distributed Systems. In: Butz, M.V., Sigaud, O., Pezzulo, G., Baldassarre, G. (eds.) *ABiALS 2006. LNCS (LNAI)*, vol. 4520, pp. 307–324. Springer, Heidelberg (2007)
5. Griffiths, G.: Enhancing Peer-to-Peer Collaboration Using Trust. *Expert Systems with Applications* 31(4), 849–858 (2006)
6. Huynh, T.D., Jennings, N.R., Shadbolt, N.R.: An Integrated Trust and Reputation Model for Open Multi-Agent System. *Autonomous Agent and Multi Agent Sys.* 13(2), 119–154 (2006)
7. Jösang, A., Ismail, R., Boyd, C.: A Survey of Trust and Reputation Systems for Online Service Provision. *Decision Support Systems* 43(2), 618–644 (2007)
8. Kim, M., Ahn, J.H.: Management of Trust in the e-Marketplace: the Role of the Buyer's Experience in Building Trust. *J. of Information Technology* 22(2), 119–132 (2007)
9. O'Donovan, J., Smyth, V.: Trust in recommender systems. In: Proc. 10th Int. Conf. on Intell. User Interfaces, pp. 167–174. ACM (2005)
10. Palopoli, L., Rosaci, D., Sarné, G.M.L.: A Multi-tiered Recommender System Architecture for Supporting E-Commerce. In: Fortino, G., Badica, C., Malgeri, M., Unland, R. (eds.) *Intelligent Distributed Computing VI. SCI*, vol. 446, pp. 71–81. Springer, Heidelberg (2013)
11. Rosaci, D.: Trust measures for competitive agents. *Knowledge-Based Systems* 28, 38–46 (2013)
12. Rosaci, D., Sarné, G.M.L.: TRES: A Decentralized Agent-Based Recommender System to Support B2C Activities. In: Hkansson, A., Nguyen, N.T., Hartung, R.L., Howlett, R.J., Jain, L.C. (eds.) *KES-AMSTA 2009. LNCS*, vol. 5559, pp. 183–192. Springer, Heidelberg (2009)
13. Rosaci, D., Sarnè, G.M.L.: Cloning Mechanisms to Improve Agent Performances. *Journal of Network and Computer Applications* 36(1), 402–408 (2013)
14. Rosaci, D., Sarnè, G.M.L.: Recommending Multimedia Web Services in a Multi-Device Environment. *Information Systems* 38(2), 196–212 (2013)
15. Rosaci, D., Sarnè, G.M.L., Garruzzo, S.: Integrating Trust Measures in Multiagent Systems. *International Journal of Intelligent Systems* 27(1), 1–15 (2012)
16. Sabater, J., Sierra, C.: REGRET: Reputation in Gregarious Societies. In: Proc. 5th Int. Conf. on Autonomous Agents, pp. 194–195. ACM (2001)
17. Sabater, J., Sierra, C.: Review on Computational Trust and Reputation Models. *Artificial Intelligence Review* 24(1), 33–60 (2005)
18. Sarvapali, D.H., Ramchurn, S.D., Jennings, N.R.: Trust in Multi-Agent Systems. *The Knowledge Engineering Review* 19, 1–25 (2004)
19. Sierra, C., Dignum, F.: Agent-Mediated Electronic Commerce: Scientific and Technological Roadmap. In: Dignum, F., Sierra, C. (eds.) *Agent Mediated Elec. Commerce 2000. LNCS (LNAI)*, vol. 1991, pp. 1–18. Springer, Heidelberg (2001)
20. van Rijsbergen, C.J.: *Information Retrieval*. Butterworth (1979)
21. Wei, K., Huang, J., Fu, S.: A Survey of E-Commerce Recommender Systems. In: Proc. of the 13th Int. Conf. on Service Systems and Service Management, pp. 1–5. IEEE (2007)

Exploration of Document Classification with Linked Data and PageRank

Martin Dostal, Michal Nykl, and Karel Ježek

Abstract. In this article, we would like to present a new approach to classification using Linked Data and PageRank. Our research is focused on classification methods that are enhanced by semantic information. The semantic information can be obtained from ontology or from Linked Data. DBpedia was used as a source of Linked Data in our case. The feature selection method is semantically based so features can be recognized by non-professional users as they are in a human readable and understandable form. PageRank is used during the feature selection and generation phase for the expansion of basic features into more general representatives. This means that feature selection and PageRank processing is based on network relations obtained from Linked Data. The discovered features can be used by standard classification algorithms. We will present promising results that show the simple applicability of this approach to two different datasets.

1 Introduction

Document classification is an important part of document management systems and other text processing services. Today's methods are usually statistically oriented, which means a large amount of data is required for the training phase of these classification algorithms. The preparation of sufficient classification training sets and

Martin Dostal
NTIS - New Technologies for the Information Society,
Faculty of Applied Sciences,
University of West Bohemia, Pilsen, Czech Republic
e-mail: madostal@ntis.zcu.cz

Michal Nykl · Karel Ježek
Department of Computer Science and Engineering,
Faculty of Applied Sciences,
University of West Bohemia, Pilsen, Czech Republic
e-mail: {nyk1m, jezek_ka}@kiv.zcu.cz

proper feature selection methods is a challenging task even for domain specialists. So the common solution to this problem is based on relatively comprehensive corpuses that contain a lot of documents divided into different classification classes. Statistical methods try to discover relations between terms and classification classes during the training phase.

Our approach recognizes interesting keywords and expands them using semantic information obtained from Linked Data. For example, based on a feature MySql we can do the feature expansion into databases without the explicit occurrence of this word in the document content. The classification phase can process these parent concepts and use them for the correct pairing of documents and classification classes.

Related work is discussed in Section 2. The basic principles of Linked Data are described in Section 3 and our approach to feature selection is described in Section 4. Evaluation of our method was performed with the 20 News Groups dataset and our collection of call for papers emails. The results are presented in Section 5.

2 Previous Work

There have been many supervised learning techniques for document classification. Some of these techniques include Naive Bayes, k-nearest neighbor, vector approaches e.g. Rocchio, support vector machines, boosting [14], rule learning algorithms [4], Maximum Entropy and Latent semantic analysis.

DBpedia was used as a source of the Linked Data presented in this article. We use a local copy of Linked Data stored in our relation database for performance purposes, but the SPARQL endpoint could also be used. DBpedia is a semantically enriched Wikipedia that was successfully employed previously for computing the semantic relatedness of documents. WikiRelate! [15] combines path-based measures, information content-based measures, and text overlap-based measures. Explicit Semantic Analysis [7] uses machine-learning techniques to explicitly represent the meaning of a text as a weighted vector of Wikipedia-based concepts.

Another approach to document classification [16] proposed a term graph model as an improved version of the vector space model [13]. The aim of this model is to represent the content of the document using the relationship between keywords. This model enables us to define similarity functions and a PageRank-style algorithm. Vectors of the PageRank score values were created for each document. The rank correlation and the term distance were used as similarity measures to assign a document to a classification class. An alternative approach to document classification uses hypernyms and other directly related concepts [2, 12]. Further improvements included feature expansion with additional semantic information based on ontology [6]. This approach [6] uses external knowledge for mapping terms to regions of concepts. For an exploration of related concepts, the traversal graph algorithm is used.

3 Linked Data

The concept of Linked Data [1] was first introduced by Tim Berners-Lee. He set up four rules for machine-readable content on the Web:

- Use URIs as names for things.
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information using the standards (RDF*, SPARQL).
- Include links to other URIs so that they can discover more things.

There are two basic problems with duplicates resources: disambiguation and co-reference resolution. These problems were discussed in [8]. DBLP and DBpedia [5] are two common Linked Data resources often used for academic research.

Linked Data contains information about a resource and moreover links to other related resources. The resources are applied as tags to documents. There are two basic types of links that we can directly use:

- Hypernym-hyponym (parent-child relation),
- Sibling-sibling (links to synonyms).

These relations are bidirectional so a child can find his parent and a parent can find his children. Relations are described by ontology predicates. Example of predicates from Linked Data: dbpedia-owl:genre, skos:broader, dcterms:subject. The meaning of these predicates differs slightly, but we can use them in the same way. An example of these relations between resources is shown in Fig. 1.

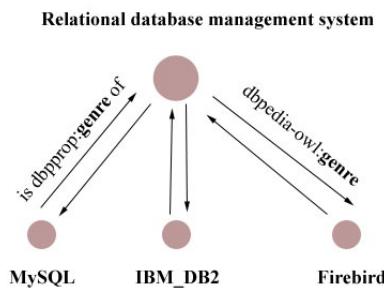


Fig. 1 Scheme of hierarchical relations between nodes in LD

Synonyms are designated by the ontology relation: owl:sameAs, which indicates true synonyms, and the relation skos:related, which indicates related concepts.

4 Feature Selection

Feature selection is the most important part of our work. Therefore, the learning phase consists of two parts in our case:

1. Training – basic features are selected and used for training in each category;
2. Features processing with Linked Data and PageRank (Fig. 2);

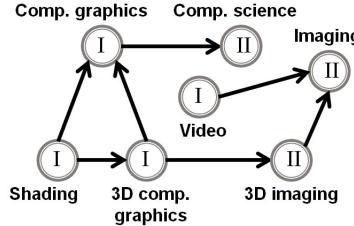


Fig. 2 Graph expansion with Linked Data and PageRank

Now we will describe the method for feature selection and their processing with Linked Data and PageRank. Basic features are selected from documents in a common way. We can use TFIDF, χ^2 or any other method. These features are mapped to nodes in Linked Data, so each feature can be identified with URI and has a clear position in a graph of Linked Data. Other features are discarded as insignificant waste. The mapping is based on full or partial compliance between the feature (term) and name of the node in the corresponding language. The nodes are usually labeled in several languages with regard to the significance and popularity of a node (Fig. 2).

The next step consists of processing these nodes with PageRank and other related nodes from Linked Data (Fig. 2). The objective of this effort is to gain related nodes with high relevance to the document, even if they were not seen in the content of the document explicitly. Graph expansion is illustrated in the Fig. 2. The original (basic) nodes are marked as I and derived nodes are marked as II. We have investigated three options for construction of the graph (see Fig. 3).

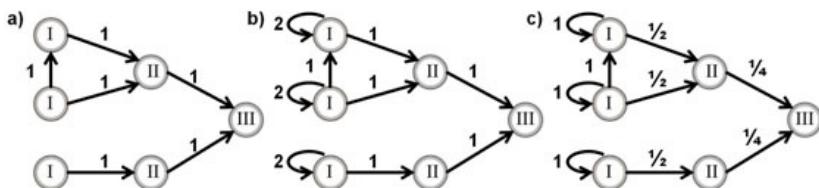


Fig. 3 Graph expansion with PageRank

- a) All edges have the same weight equal to 1.
- b) The basic nodes are advantaged with self-loop.
- c) The basic nodes are advantaged with self-loop. The edges to the derived nodes are penalized (based on the distance of the derived node from the basic node).

As expected, our evaluation of the expanded graph in Fig. 4 shows, that the variant c) achieves the best results (see Tab. 1) due to effective limitation in the expansion of the nodes. The other versions requires explicit limiting criterion for graph expansion.

Table 1 Dependency of PageRank score on nodes expansion and weights of the edges

step exp.	node	var a)			var b)		var c)
		I + II	I + II + III	I + II + III + IV	I + II	I + II + III	I + II
I	1	0.08	0.07	0.027	0.16	0.14	0.12
I	2	0.18	0.15	0.059	0.20	0.18	0.22
I	3	0.13	0.11	0.043	0.14	0.13	0.17
II	4	0.13	0.11	0.043	0.10	0.09	0.10
II	5	0.27	0.18	0.071	0.21	0.15	0.21
II	6	0.11	0.09	0.036	0.09	0.08	0.09
II	7	0.11	0.09	0.036	0.09	0.08	0.09
III	8		0.19	0.355		0.16	
IV	9			0.329			

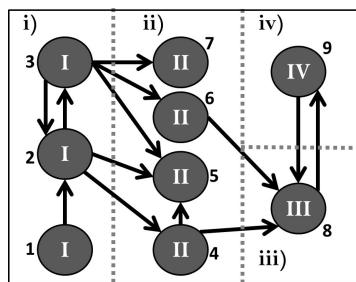


Fig. 4 Graph expansion with PageRank

Generally expressed: we begin with the nodes marked I and II. New nodes are added only if at least one of the previously added nodes got a higher the PageRank score than an arbitrary from other nodes. The higher the PageRank score, the more significant the feature is for the article. In the next section, variant c) was used for the evaluation of our feature selection approach.

5 Evaluation

Two different data collections were used for evaluation purposes: Conferences and 20 News Groups dataset [9]. Our collection of conferences consists of 15 000 calls for papers.

This collection was chosen as a basic evaluation tool. Conferences can be relatively easily divided into single and multi-topic cases. Multi-topic conferences were discarded so we were able to apply single label document classification. The remaining conferences were assigned to the corresponding topics based on the lists found on the Internet. Almost every document from this collection consists of a large number of keywords related to the category. Therefore, we were able to train categories relatively quickly. Approximately 10 solid documents were enough to train 1 category with almost constant macro-averaging $F1(\beta=1) \doteq 0.9$. This $F1$ measure was almost constant from 10 to 100 solid training documents for 1 category. After 100 training documents or 2000 features, a problem with over-training was noticed.

The 20 News groups dataset was used for the simulation of a more common case but only a subset from this collection was used. The reason was that our source of Linked Data was not sufficient to distinguish between similar categories in the 20 News groups collection like `comp.os.ms-windows.misc` and `comp.windows`. Another problem was over-training that occurs with approximately 100 training documents for 1 category.

For evaluation purposes we decided to compare our features selection method and the standard statistical approach with the same vector space classification algorithm (Rocchio). The comparison (see Fig. 5) is done with macro-averaging $F1$ measure. The number of testing documents is determined as 20% of the training documents.

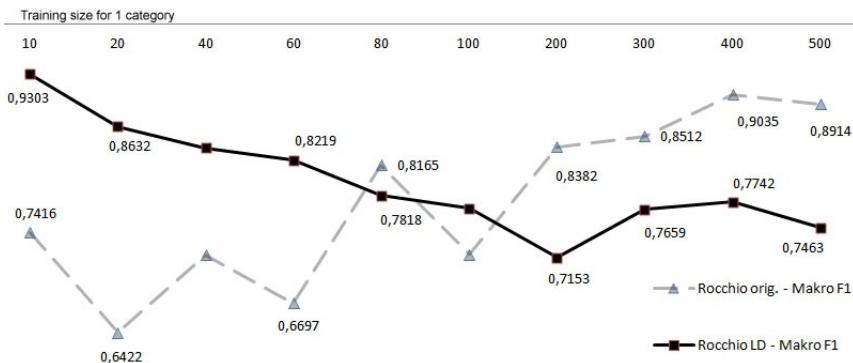


Fig. 5 Graph – $F1$ measure for Rocchio classification algorithm (the 20 News groups)

6 Conclusion and Future Work

Our method for document classification with Linked Data is promising especially in those tasks with insufficient training sets or for quick filtering of existing documents. In those cases, the training phase could be very expensive and a waste of time for

the user. Our method allows the definition of assigning categories using only a small number of training documents or a single node from Linked Data with automatic expansion on both sites - category definition and feature selection. In common tasks of document classification, the existence of keywords from interesting categories in Linked Data is required. In the future, we would like to eliminate the over-training problem and we plan to create a method for document classification directly based on the graph analysis.

Acknowledgements. This work was supported by the grants GAČR P103/11/1489, ERDF project NTIS CZ.1.05/1.1.00/02.0090 and SGS-2013-029 “Advanced computing and inf. systems”.

References

1. Berners-Lee, T.: Linked Data - Design Issues. Online document (2006), <http://www.w3.org/DesignIssues/LinkedData.html/> (Cited January 12, 2013)
2. Bloehdorn, S., Hotho, A.: Boosting for Text Classification with Semantic Features. In: Mobasher, B., Nasraoui, O., Liu, B., Masand, B. (eds.) WebKDD 2004. LNCS (LNAI), vol. 3932, pp. 149–166. Springer, Heidelberg (2006)
3. Brine, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems 30(1-7), 107–117 (1998)
4. Cohen, W., Singer, Y.: Context-sensitive learning methods for text categorization. In: Proceedings of the ACM SIGIR 1996 (1996)
5. DBpedia, <http://dbpedia.org/> (Cited January 12, 2013)
6. de Melo, G., Siersdorfer, S.: Multilingual text classification using ontologies. In: Amati, G., Carpineto, C., Romano, G. (eds.) ECiR 2007. LNCS, vol. 4425, pp. 541–548. Springer, Heidelberg (2007)
7. Gabrilovich, E., et al.: Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In: Proceedings of the IJCAI 2007, Hyderabad, India, pp. 1606–1611 (2007)
8. Jaffri, A., Glaser, H., Millard, I.: URI Disambiguation in the Context of Linked Data. In: Proceedings of the LDOW 2008, Beijing, China (2008)
9. Lang, K.: Newsweeder: Learning to filter netnews. In: Proceedings of the Twelfth International Conference on Machine Learning, 20 News groups dataset, pp. 331–339 (1995)
10. Langville, A.N., Meyer, C.D.: Google’s PageRank and Beyond: The Science of Search Engine Ranking. Princeton University Press, Princeton (2006)
11. Ma, N., et al.: Bringing PageRank to the citation analysis. Proceedings of the Information Processing & Management 44(2), 800–810 (2008)
12. Ramakrishnanan, G., Bhattacharyya, P.: Text Representation with WordNet Synsets using Soft Sense Disambiguation. In: Proceedings of the 8th NLDB, Burg, Germany (2003)
13. Salton, G.: The SMART Retrieval System. Prentice-Hall, Englewood Cliffs (1971)
14. Schapire, R., Singer, Y.: BoosTexer: A boosting-based system for text categorization. In: Machine Learning, pp. 135–168 (1999)
15. Strube, M., Ponzetto, S.P.: WikiRelate! Computing semantic relatedness using Wikipedia. In: Proceedings of the AAAI 2006, Boston, USA, pp. 1419–1424 (2006)
16. Wang, W., Do, D.B., Lin, X.: Term Graph Model for Text Classification. In: Li, X., Wang, S., Dong, Z.Y. (eds.) ADMA 2005. LNCS (LNAI), vol. 3584, pp. 19–30. Springer, Heidelberg (2005)

Matching Users with Groups in Social Networks

Domenico Rosaci and Giuseppe M.L. Sarné

Abstract. Understanding structures and dynamics of social groups is a crucial issue for Social Network analysis. In the past, several studies about the relationships existing between users and groups in On-line Social Networks have been proposed. However, if the literature well covers the issue of computing individual recommendations, at the best of our knowledge any approach has been proposed that considers the evolution of on-line groups as a problem of matching the individual users' profiles with the profiles of the groups. In this paper we propose an algorithm that addresses this issue, exploiting a multi-agent system to suitably distribute the computation on all the user devices. Some preliminary results obtained on simulated On-line Social Networks data show both a good effectiveness and a promising efficiency of the approach.

1 Introduction

Nowadays, the Internet scenario has seen a significant growth in scale and richness of On-line Social Networks (OSNs), that are becoming very complex and internally structured realities, particularly in the largest communities as Facebook, Flickr, MySpace, Google+ and Twitter. In these networks we observe an increasing diffusion of social *groups*, that are sub-networks of users having similar interests [3, 7] and sharing opinions and media contents. In the past, some studies about the relationships existing between users and groups in OSNs have been proposed. For instance, in [8], a quantitative study is presented about the influence of neighbours on the probability of a particular node to join with a group, on four popular OSNs. Moreover, the proposal presented in [1] deals with the problem of the overwhelming number of groups, that causes difficulties for users to select a right group to join. To solve this problem, the authors introduce, in the context of Facebook, the Group

Domenico Rosaci · Giuseppe M.L. Sarné

University Mediterranea of Reggio Calabria, 89123 Reggio Calabria, Italy

e-mail: {domenico.rosaci, sarne}@unirc.it

Recommendation System using combination of hierarchical clustering technique and decision tree. Also the studies presented in [2, 10, 11] deal with the problem of giving recommendations to a group of users, instead of a single user. This is a key problem from the viewpoint of creating OSN groups that provide their users with a sufficient satisfaction. The problem is not simply to suggest to a user the best groups to join with, but also to suggest to a group the best candidates to be accepted as new members. If the existing research in OSN well covers the issue of computing individual recommendations, and the aforementioned issue begins to give attention to the issue of computing group satisfaction, however at the best of our knowledge any study has been proposed to consider the issue of managing the evolution of a OSN group as a problem of matching the individual users' profiles with the profiles of the groups. The notion itself of *group profile* is already unusual in OSN analysis, although the concept of *social profile* is not new in the research area of virtual communities. For instance, in [9], following some theories originated in sociological research on communities, a model of a virtual community is presented, defined as a set of characteristics of the community.

In this paper, we provide the following contribution:

- we introduce the notion of *group profile* in the context of OSNs, giving to this notion a particular meaning coherent with the concept of OSN group. In our perspective, an OSN group is not simply a set of categories of interests, but also a set of common rules to respect, a preferred behaviour of its members, a communication style and a set of facilities for sharing media contents. Our definition of group profile is coherent with the definition of a *user profile*, that contains information comparable with those of a group profile.
- We exploit the above notion of group profile to provide each group of an OSN with a *group agent*, capable of creating, managing and continuously updating the group profile. Similarly, we associate a *user agent* with each user of the OSN.
- Likewise to other approaches we proposed in the past to build recommender systems for virtual communities [12, 14, 15, 16], introducing efficiency via the use of a distribute agent system, here we propose to exploit the agents above to automatically and dynamically computing a matching between user profiles and group profiles in a distributed fashion. We note that the idea of associating an agent as a representative of a group is not completely new in a social network scenario, having been introduced also in [4, 5, 6]. However, in these past works this idea has been exploited to allow interoperability between different groups, without facing the problem of matching users and groups. We propose to provide the user agent with a matching algorithm able to determine the group profiles that best match with the user profile. This matching algorithm, named User-to-Groups (U2G) is based on the computation of a dissimilarity measure between user and group profiles. As a result of the U2G computation, the user agent will submit on behalf of its user some requests for joining with the best suitable groups. On the other hand, the agent of each group will execute the U2G algorithm to accept, among the users that requested to join with the group, only those users having profiles that sufficiently match with the group profile. This way, the dynamic

evolution of the groups should reasonably lead to a more homogeneous intra-group cohesion.

- Some experiments we have performed on a set of simulated users and groups confirms this intuition, and also show promising efficiency a scalability of the proposed algorithm.

The remaining of the paper is organized as follows. In Section 2 we introduce our reference scenario. Section 3 present the proposed U2G matching algorithm, while Section 4 describes the experiments we have performed to evaluate our approach. Finally, in Section 5 we draw our conclusions.

2 The Social Network Scenario

In our scenario, we deal with a Social Network S , represented by a pair $\mathcal{S} = \langle U, G \rangle$, where U is a set of *users*, G is a set of *groups* of users and each group of users $g \in G$ is a subset of U (i.e., $g \subseteq U \forall g \in G$).

We assume that a single user u (resp., a single group g), of \mathcal{S} is characterized by the following properties:

- He/she (resp., it) deals with some categories of interest in the social network (e.g. *music*, *sport*, *movie*, etc.). We denote as C the set of all the possible categories of interests, where each element $c \in C$ is a string representing a given category. We denote as $INTERESTS_u$ (resp., $INTERESTS_g$) a mapping that, for each category $c \in C$, returns a real value $INTERESTS_u(c)$ (resp., $INTERESTS_g(c)$), ranging in $[0..1]$, representing the level of interest of the user u (resp., of the users of the group g) with respect to discussions and multimedia content dealing with c . The values of this mapping are computed on the basis of the actual behavior of u (resp., of the users of g).
- He/she has a preference with respect the access mode of the groups (resp., it has adopted a particular access mode). The access mode is the policy regulating the access to a group (e.g., *open*, *closed*, *secret*, etc.). We denote as $ACCESS_u$ (resp., $ACCESS_g$) the access mode (represented by a string) associated with u (resp., g).
- He/she adopts or does not adopt (resp., it tolerates or does not tolerate) some possible *behaviour* available in the social network. A behaviour is a type of action that a user could perform, e.g. “publishing more than 2 posts per hour”, or “publishing posts longer than 500 characters”. We suppose each possible behaviour is represented by a boolean variable, that is equal to *true* if this behaviour is adopted, *false* otherwise. We denote as $BEHAVIOURS_u$ (resp., $BEHAVIOURS_g$) a set of behaviours, representing how u (resp., g) behaves with respect to all the possible behaviours. For instance, if $BEHAVIOURS = \{b_1, b_2\}$, where b_1 represents the behaviour of publishing more than 2 posts per hour, and b_2 represents the behaviour of publishing in most of the cases posts longer than 500 characters, then a property $BEHAVIOURS_u = \{\text{true}, \text{false}\}$ characterizes a user u that publishes more than 2 posts per hour and that, in the most of cases, does not publish posts longer than 500 characters.

- He/she has (resp., its users have) a set of *friends*, that are users of the social network. We denote as FRIENDS_u (resp., FRIENDS_g) the set of all the users that are friends of u (resp., the set of all the users that are friends of at least a member belonging to the group g).

Then, we define the profile p_u (resp., p_g) of a user u (resp., a group g) as a tuple $\langle \text{INTERESTS}_u, \text{ACCESS}_u, \text{BEHAVIOURS}_u, \text{FRIENDS}_u \rangle$ (resp., $\langle \text{INTERESTS}_g, \text{ACCESS}_g, \text{BEHAVIOURS}_g, \text{FRIENDS}_g \rangle$). Moreover, we assume that a software agent a_u (resp., a_g) is associated with each user u (resp., group g). The agent a_u (resp., a_g) automatically performs the following tasks:

- it updates the profile p_u (resp., p_g) of its user (resp., group) each way the user u (resp., a user of the group g) performs an action involving some information represented in p_u (resp., p_g). In particular, each time the user u publishes a post, or comments an already published post, dealing with a category c , the new value $\text{INTERESTS}_u(c)$ is updated as follows:

$$\text{INTERESTS}_u(c) = \alpha \cdot \text{INTERESTS}_u(c) + (1 - \alpha) \cdot \delta$$

that is a weighted mean between the previous interest value and the new value, where α and δ are real values (ranging in $[0..1]$). More in detail, δ represents the increment we want to give to the u 's interest in c consequently of the u 's action, while α is the importance we want to assign to the past values of the interest value with respect to the new contribution. The values α and δ are arbitrarily assigned by the user itself. Similarly, the $\text{INTERESTS}_g(c)$ value of a group g is updated by the agent a_g each time the $\text{INTERESTS}_u(c)$ value of any user $u \in g$ changes. The new value of $\text{INTERESTS}_g(c)$ is computed as the mean of all the $\text{INTERESTS}_u(c)$ values $\forall c \in g$. Moreover, each way the user u performs an action in the social network (e.g. publishing a post, a comment, etc.) its agent a_u analyses the action and consequently sets the appropriate boolean values for all the variables contained in BEHAVIOURS_u (e.g., if BEHAVIOURS_u contains a variable representing the fact of publishing more than 2 posts per hour, then a_u checks if the action currently performed by u makes true or false this fact and, consequently, sets the variable).

Analogously, the agent a_g , associated with a group g , updates the variables contained in BEHAVIOURS_g each time the administrator of g decides to change the correspondent rules (e.g., e.g. if BEHAVIOURS_g contains a variable representing the fact of publishing more than 2 posts per hour, and the administrator of g decides that this fact is not tolerated in the group, then a_g sets the correspondent variable to the value *false*).

Furthermore, if the user u (resp., the administrator of the group g) decides to change his/her preference with respect to the access mode, the agent a_u (resp., a_g) consequently updates ACCESS_u (resp., ACCESS_g). Finally, if the user u (resp., a user of the group g) adds a new friend in his/her friends list, or deletes an assisting friend from his/her friends list, the agent a_u (resp., a_g) consequently

- updates FRIENDS_u (resp., FRIENDS_g). Note that the agent a_g computes the property FRIENDS_g as the union of the sets FRIENDS_u of all the users $u \in g$.
- Periodically, the agent a_u (resp., a_g) executes the user agent task (resp., group agent task) described in Section 3, in order to contribute to the *User-to-Group (U2G) Matching* global activity of the social network.

In order to perform the above tasks, each agent can interact with each other, sending and receiving messages. This possibility is assured by the presence of a *Directory Facilitator* agent (DF), associated with the whole social network, that provides a service of Yellow Pages. More in particular, the names of all the users and groups of the social network are listed in an internal repository of the DF, associating with each user and group the corresponding agent name. A *Communication Layer* allows an agent x to send a message to another agent y simply by using the name of y in the *receiver* field of the message. Note that maintaining the DF naming repository is the only centralized activity in our social network scenario, while the algorithm computing the U2G matching is completely distributed on the whole agent network.

3 The U2G Matching Algorithm

In our scenario, the U2G matching is a global activity distributed on the user and group agents belonging to the agent network. More in particular, each user agent a_u (resp. group agent a_g) periodically executes the following *user agent task* (resp. *group agent task*), where we call *epoch* each time the task is executed, and we denote as T the (constant) period between two consecutive epochs.

3.1 The User Agent Task

Let X be the set of the n groups the user u is joined with, where $n \leq n_{NMAX}$, being $NMAX$ the maximum number of groups which a user can join with. We suppose that a_u : (i) records into an internal cache the profiles of the groups $g \in X$ obtained in the past by the associated group agents; (ii) associates with each profile p_g the date of acquisition, denoted as date_g . Let also m be the number of the group agents that at each epoch must be contacted by a_u . In such a situation, a_u behaves as follows:

- In the DF repository, it randomly selects m groups that are not present in the set X . Let Y be the set of these selected groups, and let $Z = X \cup Y$ a set containing all the groups present in X or in Y .
- For each group $g \in Y$, and for each group $g \in X$, such that the date of acquisition date_g is higher than a fixed threshold ψ , it sends a message to the group agent a_g , whose name has obtained by the DF, requesting it the profile p_g associated with the group.
- For each received p_g , it computes a *dissimilarity measure* between the profile of the user u and the profile of the group g , defined as a weighted mean of four contributions c_I, c_A, c_B and c_F , associated with the properties *INTERESTS*, *ACCESS*, *BEHAVIOURS* and *FRIENDS*, respectively. More in particular, each

of this contribution measures how much are different the values of the corresponding property in p_u and in p_g . To this purpose:

- c_I is computed as the average of the differences (in the absolute value) of the interests values of u and g for all the categories present in the social network, that is:

$$c_I = \frac{\sum_{c \in C} |INTERESTS_u(c) - INTERESTS_g(c)|}{|C|}$$

- c_A is set equal to 0 (resp., 1) if $ACCESS_u$ is equal (resp., not equal) to $ACCESS_g$.
- c_B is computed as the average of all the differences between the boolean variables contained in $BEHAVIOURS_u$ and $BEHAVIOURS_g$, respectively, where this difference is equal to 0 (resp., 1) if the two corresponding variables are equal (resp., different).
- c_F is computed as the percentage of common friends of u and g , with respect to the total numer of friends of u or g . That is:

$$c_F = \frac{|FRIENDS_u \cap FRIENDS_g|}{|FRIENDS_u \cup FRIENDS_g|}$$

Note that each contribution has been normalized in the interval $[0..1]$, for making comparable all the contributions. The dissimilarity d_{ug} of a group g with respect to the user u is then computed as:

$$d_{ug} = \frac{w_I \cdot c_I + w_A \cdot c_A + w_B \cdot c_B + w_F \cdot c_F}{w_I + w_A + w_B + w_F}$$

- Now, let τ a real value, ranging in $[0..1]$, representing a threshold for the dissimilarity, such that each group $g \in Z$ is considered as a good candidate to join if (ii) $d_{ug} < \tau$ and (ii) it is inserted by a_u in the set $GOOD$. Note that if there exist more than $NMAX$ groups satisfying this condition, the $NMAX$ groups having the smallest values of global difference are selected. For each selected group $g \in GOOD$, when $g \notin X$, the agent a_u sends a join request to the agent a_g , that also contains the profile p_u associated with u . Otherwise, for each group $g \in X$, when $g \notin GOOD$, the agent a_u leaves the group g .

3.2 The Group Agent Task

Let K be the set of the k users joined with the group g , where $k \leq n_{KMAX}$, being $KMAX$ the maximum number of members allowed by the group administrator. We suppose that a_g stores into an internal cache the profiles of the users $u \in K$ obtained in the past by the associated user agents, and also associates with each profile p_u the date of acquisition, denoted as $date_u$. Each time a_g receives a join request by a user agent r , that also contains the profile p_r associated with r , it behaves as follows:

- For each user $u \in K$ such that the date of acquisition $date_u$ is higher than a fixed threshold η , it sends a message to the user agent a_u , whose name has obtained by the DF, requesting it the profile p_u associated with the user.
- After the reception of the responses from the contacted user agents, it computes the *dissimilarity measure* d_{ug} between the profile of each user $u \in K \cup \{r\}$ and the profile of the group g , following the definitions given in Section 3.1.
- Now, let π a real value, ranging in $[0..1]$, representing a threshold for the dissimilarity, such that a user u is considered as acceptable to join if $d_{ug} < \pi$. Then, the agent a_g stores in a set $GOOD$ those users $u \in K \cup \{r\}$ such that $d_{ug} < \pi$ (if there exist more than $KMAX$ users satisfying this condition, the $KMAX$ users having the smallest values of global difference are selected). If $r \in GOOD$, a_g accepts its request to join with the group. Moreover, for any user $u \in K$, with $u \notin GOOD$, a_g deletes u from the group.

4 Experiments

In this section, we describe some preliminary experiments we performed to evaluate the effectiveness of the U2G matching activity in making more homogeneous the groups of an OSN. To this purpose, we have built an OSN simulator, written in JAVA and based on the JADE platform, capable of simulating the users' behaviours and the activities of their associated agents, as well as the group administrators' behaviour and the activities of the group agents, in a social network. In our simulation, we considered an OSN with 150.000 users and 1200 groups. The simulator provided each user and each group with a user profile, having the structure described in Section 2. More in detail, the profile p_u of a user u is generated as follows:

- each values $INTERESTS_u(c)$ is a random value from a uniform distribution of values in $[0..1]$;
- $ACCESS_u$ is assigned from three possible values, namely *OPEN*, *CLOSED* and *SECRET*, such that the probability of assigning the value *OPEN* (resp., *CLOSED*, *SECRET*) is set to 0.7 (resp., 0.2, 0.1). This distribution values appear reasonable in a realistic OSN scenario;
- $BEHAVIOURS_u$ contains six boolean variables, representing the user's attitude to: (i) publish more than 1 post per day; (ii) publish posts longer than 200 characters in most of the cases; (iii) publish at least two comments per day to posts of other users; (iv) respond to comments associated with its posts in most the cases; (v) leave at least 2 rates "I like it" per day; (vi) respond to a message of another user in most the cases. The values of these variables are randomly generated from a uniform distribution on the possible value-set $\{true, false\}$;
- in $FRIENDS_u$, the simulator inserts a set of other users, randomly choosing one of the following distributions: (i) users that have a dissimilarity with u smaller than 0.5 (the dissimilarity is computed in the same way of d_{ug} in Section 2); (ii) users randomly chosen from the set of the OSN users; (iii) 50 percent of the users generated as in (i) and the remaining 50 percent generated as in (ii). These distributions represent three realistic types of users, namely those that select their

Table 1 Values used for the U2G parameters

	<i>alpha</i>	δ	ψ (days)	τ	η (days)	π
<i>Value</i>	0.5	0.1	10	0.7	10	0.7

friends based on similar preferences and behaviour, those that randomly accept any friendship and those that behaves in an intermediate fashion with respect to the first two attitudes.

Users are then randomly assigned to the available groups, in such a way that a user is joined at least with 2 groups and at most with 10 groups. The profile p_g of each group g is assigned generating completely random values for the properties $ACCESS_g$ and $BEHAVIOURS_g$, while the properties $INTERESTS_g$ and $FRIENDS_g$ are computed based on the corresponding members' values, following the formulas described in Section 2. The values of the parameters introduced in Section 3 are shown in Table 1.

As a measure of the internal *cohesion* of a group, we use the concept of *average dissimilarity*, commonly exploited in Clustering Analysis [13], defined as the average of the dissimilarities between each pair of objects in a cluster. In our scenario, a group g is the equivalent of a cluster of users, and the average dissimilarity of g , denoted as AD_g is computed as $\frac{\sum_{x,y \in g, x \neq y} d_{xy}}{|g|}$.

In order to measure the global cohesion of the groups of the social network, we compute the mean MAD and standard deviation DAD of all the AD_g , by the formulas: $MAD = \frac{\sum_{g \in G} AD_g}{|G|}$; $DAD = \sqrt{\frac{\sum_{g \in G} (AD_g - MAD)^2}{|G|}}$

In our simulation, after the random generation of the profiles of users and groups, the initial values for the above measures were $MAD = 0.512$ and $DAD = 0.043$, indicating a population with a very low homogeneity, due to the completely random generation of the groups. Starting from this initial configuration, we have applied the U2G algorithm described in Section 3, simulating a number of 150 epochs of execution per each user, where each epoch simulated a time period of a day. The results of the simulation, in terms of MAD and DAD with respect to the epochs, are shown in Figure 1-A and 1-B. The results clearly show that the U2G algorithm introduces a significant increment of the cohesion in social network groups, that after a period of about 110 epochs achieves a stable configuration represented by $MAD = 0.163$ and $DAD = 0.0095$. Moreover, we repeated the experiments above, changing the number of simulated users and groups. In particular, in Figure 1-(C) we have plotted the stable MAD/DAD for different values of the users' number, having fixed to 1200 the number of the groups, while in Figure 1-(D) the stable MAD/DAD values are reported for different values of the groups' number, having fixed to 150.000 the number of the users. The results show that the number of the necessary epochs for achieving a stable configuration increases almost linearly with respect to the number of the groups, confirming a good scalability of the approach.

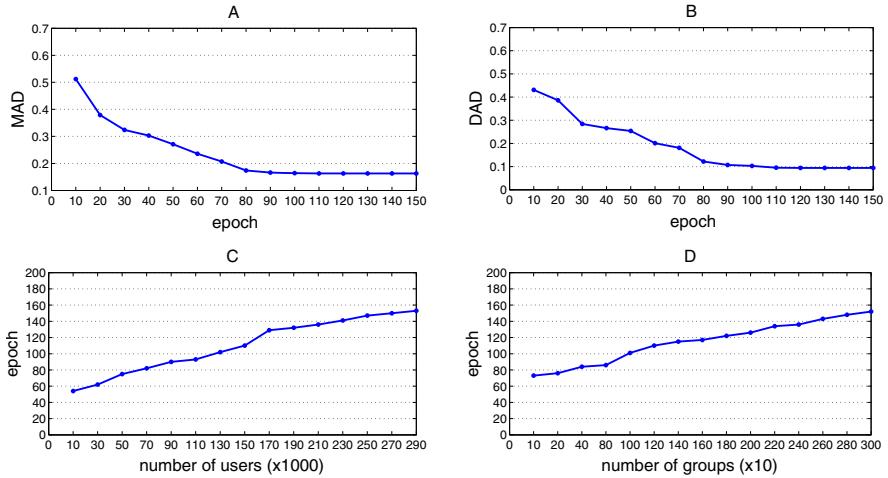


Fig. 1 The variation of (A) MAD and (B) 10*DAD vs epochs and the variation of the epoch corresponding to stable MAD/DAD for (C) different number of users and 1200 groups and (D) different number of groups and 150.000 users

5 Conclusion

The problem of making possible a suitable evolution in OSN groups, dynamically increasing the intra-group cohesion, is emerging as a key issue in the OSN research field. If the notion of homogeneity is becoming already more complex with the introduction of high-structured user profiles, leading to design sophisticated approaches for computing similarity measures, on the other hand the large dimensions of current OSNs, as well as the continuously increasing number of groups, introduce the necessity of facing efficiency and scalability problems. In this paper, we present a User-to-Group matching algorithm, that allows a set of software agents associated with the OSN users to dynamically and autonomously manage the evolution of the groups, detecting for each user the most suitable groups to join with based on a dissimilarity measure. Moreover, the agents operate on behalf of the group administrators, such that a group agent accepts only those join requests that come from users profile compatible with the profile of the group. Some preliminary experiments clearly show that the execution of the matching algorithm increases in time the internal cohesion of the groups composing the social network.

Acknowledgements. This work has been partially supported by the TENACE PRIN Project (n. 20103 P34XC) funded by the Italian Ministry of Education, University and Research.

References

1. Baatarjav, E.-A., Phithakkitnukoon, S., Dantu, R.: Group Recommendation System for Facebook. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2008 Workshops. LNCS, vol. 5333, pp. 211–219. Springer, Heidelberg (2008)
2. Basu Roy, S., Amer-Yahia, S., Chawla, A., Das, G., Yu, C.: Space Efficiency in Group Recommendation. *The VLDB Journal* 19(6), 877–900 (2010)
3. Buccafurri, F., Rosaci, D., Sarné, G.M.L., Palopoli, L.: Modeling Cooperation in Multi-Agent Communities. *Cognitive Systems Research* 5(3), 171–190 (2004)
4. De Meo, P., Nocera, A., Quattrone, G., Rosaci, D., Ursino, D.: Finding Reliable Users and Social Networks in a Social Internetworking System. In: Proc. of the 2009 Int. Database Engineering & Applications Symp., pp. 173–181. ACM (2009)
5. De Meo, P., Nocera, A., Rosaci, D., Ursino, D.: Recommendation of Reliable Users, Social Networks and High-Quality Resources in a Social Internetworking System. *AI Communications* 24(1), 31–50 (2011)
6. De Meo, P., Quattrone, G., Rosaci, D., Ursino, D.: Dependable Recommendations in Social Internetworking. In: Web Intelligence and Intelligent Agent Technologies, pp. 519–522. IET (2009)
7. Gauch, S., Speretta, M., Chandramouli, A., Micarelli, A.: User Profiles for Personalized Information Access. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) *The Adaptive Web*. LNCS, vol. 4321, pp. 54–89. Springer, Heidelberg (2007)
8. Hui, P., Buchegger, S.: Groupthink and Peer Pressure: Social Influence in Online Social Network Groups. In: Int. Conf. on Advances in Social Network Analysis and Mining, ASONAM 2009, pp. 53–59. IEEE (2009)
9. Hummel, J., Lechner, U.: Social Profiles of Virtual Communities. In: Proc. of the 35th Annual Hawaii Int. Conf. on System Sciences, HICSS 2002, pp. 2245–2254. IEEE (2002)
10. Kasavana, M.L., Nusair, K., Teodosic, K.: Online Social Networking: Redefining the Human Web. *Journal of Hospitality and Tourism Technology* 1(1), 68–82 (2010)
11. Kim, J.K., Kim, H.K., Oh, H.Y., Ryu, Y.U.: A Group Recommendation System for Online Communities. *Int. Journal of Inf. Management* 30(3), 212–219 (2010)
12. Palopoli, L., Rosaci, D., Sarné, G.M.L.: A Multi-Tiered Recommender System Architecture for Supporting E-Commerce. In: Fortino, G., Badica, C., Malgeri, M., Unland, R. (eds.) *Intelligent Distributed Computing VI*. SCI, vol. 446, pp. 71–81. Springer, Heidelberg (2013)
13. Pearson, R.K., Zylkin, T., Schwaber, J.S., Gonye, G.E.: Quantitative Evaluation of Clustering Results Using Computational Negative Controls. In: Proc. 2004 SIAM Int. Conf. on Data Mining, pp. 188–199 (2004)
14. Rosaci, D., Sarné, G.M.L.: Recommending Multimedia Web Services in a Multi-device Environment. *Information Systems* 38(2), 198–212
15. Rosaci, D., Sarné, G.M.L.: Efficient Personalization of e-Learning Activities Using a Multi-Device Decentralized Recommender System. *Computational Intelligence* 26(2), 121–141 (2010)
16. Rosaci, D., Sarné, G.M.L.: A multi-agent recommender system for supporting device adaptivity in e-Commerce. *Journal of Intelligent Information Systems* 38(2), 393–418 (2012)

Semantically Partitioned Peer to Peer Complex Event Processing

Filip Nguyen, Daniel Tovarňák, and Tomáš Pitner

Abstract. Scaling Complex Event Processing applications is inherently problematic. Many state of the art techniques for scaling use filtering on producers, vertical scaling, or stratification of an Event Processing Network. The solutions usually aren't distributed and require centralized coordination. In this paper, we are introducing a technique for scaling Complex Event Processing in a distributed fashion and by taking semantic information of events into account. We are introducing two CEP models for scaling CEP architectures, providing core algorithms, and evaluating their performance.

1 Introduction

In this paper, we are concerned with a new way of scaling Complex Event Processing (CEP) applications. This section introduces CEP, presents the motivation for our work, and introduces our contribution.

1.1 Complex Event Processing

CEP is both a theoretical and a practical research area that studies events and event processing in current computing systems and businesses. Examples of such events may be:

- A payment using a credit card. This may be regarded as relatively infrequent event.
- A barcode reading of a product code. This may be regarded as a frequent event.
- A motion sensor on electronic doors to a supermarket.

Filip Nguyen · Daniel Tovarňák · Tomáš Pitner
Faculty of Informatics, Masaryk University,
Brno, Czech Republic
e-mail: {xnguyen, xtovarn, tomp}@fi.muni.cz

All of these events may be correlated together and may thusly cross both technological and domain boundaries. The process of correlating is referred to as *pattern matching*. Such a pattern might be: two payments made by the same credit card in different supermarkets within a time frame of 1 hour.

Another possible view of Complex Event Processing is an upside down version of standard data processing. Instead of data sitting in a database, waiting for queries to be submitted, CEP may be viewed as queries sitting and waiting for data to be submitted. That is why CEP is used as a monitoring tool ([4], [5]). An interesting method of using CEP as a monitoring was introduced in [2], where it is usable even for data mining.

Motivation for CEP nicely correlates with the current advent of Big Data, which is a data centric approach to extracting meaningful data. The problem here is not to store such data, but to retrieve it and to extract meaningful information [1]. CEP is able to carry out extractions of data regarding time, a frequent component in data. In CEP, data that is not processed is simply discarded.

CEP was motivated and introduced in a very comprehensive publication [3]. The book provides a basic framework for many CEP usages, including dynamic complex event processing. The book introduces the so called *Event Processing Agent* (EPA). EPA is defined as an object that monitors events to detect certain patterns in them. Another abstraction is then introduced in the form of an *Event Processing Network* (EPN) which connects many EPAs into a network where each EPA sends events to another EPA. EPNs are added to the model for flexibility and allow for the easy reuse of EPAs and for the building of dynamic EPNs. In these dynamic EPNs, processing agents may enter or leave the network at the time of processing.

1.2 Peer to Peer Complex Event Processing

We exploit the fact that events from certain producers are related to each other in some way. We do this by breaking up the event space by identifying related producers using CEP itself. We then exploit this information and put processing only where it is most needed. For example, knowing that two retail stores were visited by the same customer in the last two days signifies some connection between the two retail stores. With information that the producers are related, we are able to add event processing between these two retail stores. This event processing between these two stores may consider much more fine grained events. By using this approach we may lose some information because fine grained analysis are not done everywhere. A similar probabilistic view of CEP was also discussed in [6] using an example of a hypothetical surveillance system that is monitored using CEP.

To show how we help scale CEP, we are introducing two graph oriented event processing abstractions: *peer model* and *centralized model*. In both models, we view a graph of CEP engines and producers as a basic building block for creating scalable event processing architecture.

To uncover possible correlations between producers, we use *partitioning algorithms*. These two algorithms help to identify which producers should have the engine among them (using the centralized model). We later also map this centralized approach to the peer model.

We have developed two partitioning algorithms. Their properties are discussed and they are studied on an experimental basis in a simulated CEP environment.

2 State of the Art

In this chapter we are introducing current models of Complex Event Processing that relate to our research. The word "distributed" is being attached to CEP in many publications, but its meaning varies. In [3] the interpretation is: the processing of heterogeneous events, generated from various sources in an enterprise environment. In this environment, events are generated very often and they travel via various channels (a private computer network, the Internet) to be processed.

There are several approaches to scaling CEP. A straightforward way is by analyzing bottlenecks in current CEP systems. The authors of [9] did extensive profiling of the Borealis CEP system and identified that communication is one of the biggest bottlenecks. Another approach was introduced in [10] where traditional query planning was applied to CEP and optimizations. Sliding window push downs were successfully applied. Scalability via hierarchy was introduced in [2].

Stratification is a particularly interesting way of scaling CEP. It was introduced in [12]. Stratification significantly improves event processing capabilities by parallelizing certain matching operations. The input for a stratification algorithm is EPN with dependency information. Dependency information states which EPA depends on the input of another EPA. Using a simple algorithm, the authors were able to cleanly separate EPAs into sets called strata, which contain agents that can run in parallel.

In this paper, we understand distributed CEP as distributed collection, distributed processing and most importantly, distributed matching of events.

3 Peer to Peer Complex Event Processing

In this section we will introduce our CEP models. We will also discuss the *event space correlation problem* as well as our approach to solve it.

Our notation has two parts: graphical representation and query language.

First we will look at graphical representation. Figure 1 shows the basic building blocks of our model. The black node represents the centralized engine that accepts events. The producers of the events are represented by red nodes with light rings. The producers only produce events and do nothing else. Red nodes with bold rings



Fig. 1 CEP Graph Model

represent peers. A peer is a CEP engine that is placed directly on a producer and replaces the centralized engine in our model. Our notation further includes an event which may be augmented with time information pertaining to its creation as well as with the name of the producer.

In our model, we denote the set of producers, event engines, and peers using capital P , e.g. P_1, P_2, \dots, P_N . Set of events is denoted E and discrete events may be denoted by syntax $name : time : producer$. Events are described using two functions: $time : E \rightarrow N$ for the time when the E originated and $producer : E \rightarrow P$ to get the producer of the event. The producers may be connected in an undirected graph and a producer may send some of his events to his neighbor. The function $events : P \times P \rightarrow 2^E$ is used to retrieve a set of events that have traveled between the two producers. The event engine is a node in our graph that, upon reception of an event (E_0 signifies incoming event), tries to use this event to match a pattern. Patterns MP are represented by a simple SQL like syntax, e.g. WHERE $E_0=E_1$ WINDOW(10). The WINDOW part of SQL syntax denotes the time frame in which the matching takes place.

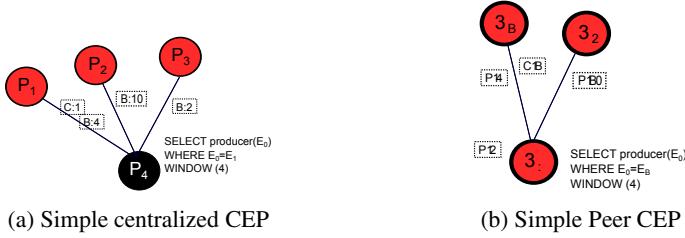
Function $patterns : P \rightarrow MP$ returns subset of patterns running on the peer.

Function $matches : MP \times N \rightarrow 2^E$ returns matched set of events matched by a pattern since specified time.

There are two possible ways to model an event processing situation using these building blocks: peer model and centralized model. Centralized model uses centralized engines and producers. In this case, events flow from the producers to the engines. Figure 2a shows a simple CEP network where producers P_1, P_2, P_3 send their events to P_4 . The P_4 in this example contains only one matching rule. This rule fires only once when event $B : 4$ arrives to P_4 . The $B : 10$ doesn't match the rule because the sliding WINDOW is only 4 time units in this case.

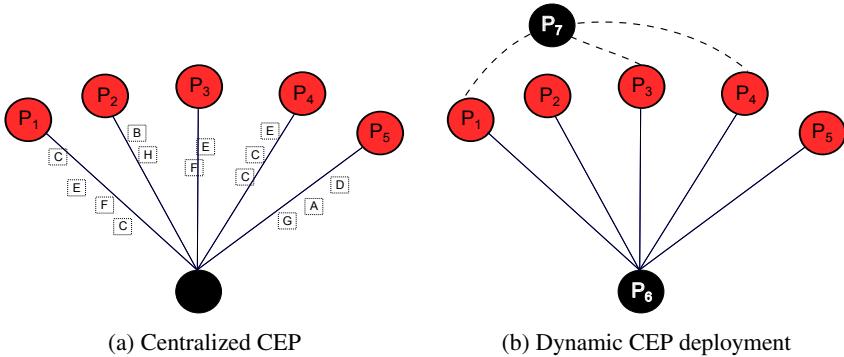
The second possible way to model an event processing situation is *peer model*. The basic building block is peer: both a producer and an event processing engine. In this peer model, a graph is formed only by peers and it is possible to achieve the same matching capabilities as with the centralized model. In Figure 2b, you can see an example of peer model notation. P_3 works both as the producer and the engine that works instead of P_4 . Note that P_3 also produces events back to itself.

Using the centralized model is better when explaining the concepts. On the other hand, the peer model resembles our targeted implementation more as well as introduces additional advantages.

**Fig. 2** CEP Models Comparison

3.1 Event Space Correlation Problem

Suppose we are building a CEP solution for 5 producers which are producing events. Figure 3a shows such a situation. Suppose we want the engine to answer to the following query: $\text{SELECT } \text{producer}(E_0), \text{producer}(E_1) \text{ WHERE } E_0 = E_1 \text{ WINDOW}(10)$.

**Fig. 3** CEP Models Comparison

Consider the technological implementation of P_6 . To be able to match the equality, the CEP engine needs to accumulate all of the events in the time frame and continually analyze whether a new event is correlating with any of the other events thus far accumulated. Because all of the events are related to each other, no stratification is possible here. This problem is common among CEP engines and was also discussed in [2], where the authors used rule allocation algorithms in their distributed CEP model. They considered a structure of rules to circumvent the problem. A different approach to solving a special case of this problem is to design a special data structure to hold shared events [8].

We have exploited the observation that producers are related. In the given example on Figure 3a, we can see that producers P_1, P_2, P_4 are correlating together more

than the other producers. The fact that these 3 are related in some way can be exploited by placing a new P_7 engine between them. This engine will have to consider a lower number of events and thus its performance will also be better.

Methods for detecting some related producers are discussed further in this paper. We call these methods *partitioning algorithms*.

In this section we discussed the dynamic placement of engines between producers and we made use of a classic centralized CEP model. This facilitates easy understanding, but it is important to note how this can be reinterpreted into our peer model. In the peer model, we do not deploy additional engines, but we only connect the appropriate engines and assign roles. Figure 4 shows an example of peer model. The peer P_5 in this situation assumes the role of P_6 from the previous example. When our algorithms decide that P_1, P_2, P_3 are related, the three peers connect (dashed line) and P_3 takes responsibility of correlating events between them. Let us portray the example in real world scenarios.

One example might be the detection of network attacks [13]. Computers and network elements are the producers of network logging information. It is possible, by monitoring coarse grained events in a network, to find out that some limited set of computers is vulnerable to attack. Using this knowledge, we can deploy an additional engine among the limited number of producers and process more fine grained events.

Another example would be CEP engine used for monitoring public transportation [14]. Sensors in vehicles are producers. By detecting the coarse grained event of "user comfort by temperature", we can detect that some vehicles are good candidates to be closely analyzed and we can deploy an additional engine in the vehicle computer itself and analyze more fine grained events (speed changes, vehicle technical data, time table precision of the driver).

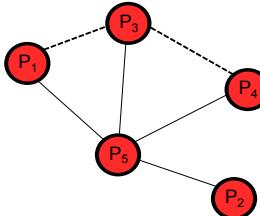


Fig. 4 Peer Model

The peer architecture of Complex Event Processing has some obvious downsides. In the real world, it may be challenging for a producer to be able to see every other producer, e.g. every supermarket server may not have access to every other supermarket server. Some event patterns might not be detected when the partitioning algorithm doesn't connect the correct producers.

However, there are many advantages of this architecture. The previously mentioned scaling capabilities. It is easy to dynamically connect several nodes and

thusly begin processing among them. Also, because producers feature an event processing engine, it is possible to easily filter events. In addition to that each added producer will improve the power of the whole system because of the addition of another processing engine.

3.2 Partitioning Algorithms

We are introducing and comparing two partitioning algorithms: the CEP based algorithm and the Monte Carlo Partitioning Algorithm. The algorithms dynamically add/remove CEP engines among peers to conduct pattern matching.

The CEP based algorithm uses complex event processing of coarse grained events (less frequent, not consuming too many resources) to later deploy more fine grained events. The present matching patterns are configurable by the user and should be prepared for a specific purpose. The algorithm uses *centralized model*.

We have used **wait** in the algorithms, which signifies waiting either for a specific guard or to wait for a specified time. While this wait is in progress, matching continues in the CEP environment. This waiting happens only to the partitioning algorithm.

The first algorithm has matching patterns as an input (MP_1, MP_2, MP_3), together with a graph representation of producers (P) and edges between them (channels). The algorithm divides the producers into two subsets (P_{left} and P_{right}) and continually checks whether the division into these two subsets is valid. To do the check it uses *differ significantly* - this check can be substituted to any configurable set difference algorithm. We use a simple set difference with threshold of 25% difference.

```
INPUT: COARSE GRAINED  $MP_1$ . FINE GRAINED  $MP_2, MP_3$ . CEP GRAPH  $G(P, channels)$ 
2.  $channels := P_c \times P$ ;  $patterns(P_c) = \{MP_1\}$ ;  $T := 0$ ;
3. WAIT UNTIL  $|\{p | \exists e \in matches(MP_1, T). p \in producer(e)\}| >= |P|/2$ 
    $X := \{p | \exists e \in matches(MP_1, T). p \in producer(e)\}$ 
    $T := CURRENT\ TIME$ 
    $channels := channels \cup \{(x, P_{left}) | x \in P\} \cup \{(x, P_{right}) | x \in P/X\}$ 
    $patterns(P_{left}) := \{MP_2\}$ 
    $patterns(P_{right}) := \{MP_3\}$ 
4. SAME WAIT AS IN (3)
   IF  $\{p | \exists e \in matches(MP_1, T). p \in producer(e)\}$  DIFFER SIGNIFICANTLY FROM X
      GOTO (2)
   ELSE
      GOTO (4)
```

The Monte Carlo algorithm uses a centralized engine at the beginning. It derives the interesting producers and correlates with the current set of matching rules. It then divides the event space. A big advantage of this algorithm is that it doesn't need any input from the the CEP user. It works with matching patterns which have already been provided for the centralized version of CEP. The algorithm divides producers to two sets SH and SL based on the vector $STAT$.

```
INPUT: EXISTING MATCHING PATTERNS  $MP$ .  $T := 0$ . CEP GRAPH  $G(P, channels)$ 
1. RANDOMLY SELECT  $MP_{small} \subset MP$ ;
2.  $channels := P_c \times P$ ;  $patterns(P_c) = MP_{small}$ ;  $T := 0$ 
```

```

3. WAIT DEFINED TIME. THEN  $\forall x \in matches(MP_{small}, T)$ 
    STAT[producer(x)] += 1
    LET SH, SL BE SET OF PRODUCERS.  $|SH| - |SL| < 1 \wedge \forall x \in SH, \forall y \in SL. STAT[x] \geq STAT[y]$ 
    channels := channels  $\cup \{(x, P_{left}) | x \in SH\} \cup \{(x, P_{right}) | x \in SL\}$ 
    patterns( $P_{left}\rangle := MP$ ; patterns( $P_{right}\rangle) := MP$ 
4. T := CURRENT TIME; CONSTRUCT STAT2 SAME WAY AS IN (2)
    IF STAT2 FROM STAT DIFFERS SIGNIFICANTLY
        GOTO (2)
    ELSE
        GOTO (4)

```

3.3 Evaluation

We have implemented and measured these two algorithms in a simulated event processing environment. This environment consisted of 100 event producers, each producing up to 10 000 event types at random times from a non-uniform distribution. The experiment was divided into two time periods.

In the first period, a random 20% of the producers were related. They produced only 1 000 event types.

In the second time period, a different 20% of producers were chosen to be related.

As we can see from Figure 5a, the Monte Carlo is even better than ideal pattern matching at a point. This is due to the fact that the two engines operating in the parallel display had an overall better performance than the one overloaded ideal engine - from time to time.

Figure 6 shows that ideal CEP engine has significantly higher matching capabilities over the CEP based solution. While disturbing, after deeper thought this is understandable. The CEP Based solution relies on the user to define the rule for dividing the producers. In a simulated environment this is hard to do, however we believe that this performance will be improved in more realistic scenarios.

From Figure 7, we can see that both algorithms drop in matching performance when different events become related. However, the Monte Carlo is better in overall matching capabilities. On the other hand, the CEP based algorithm provides better

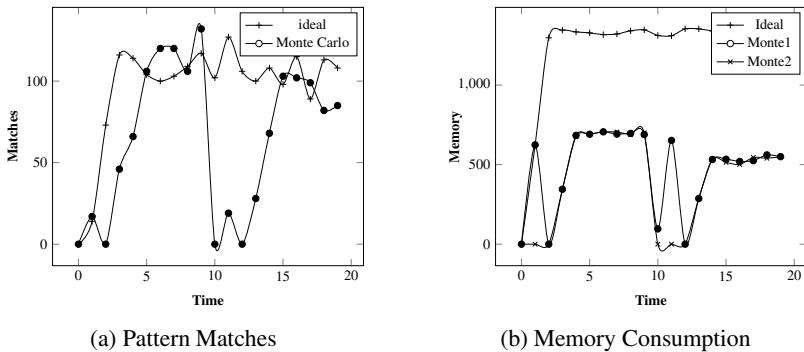
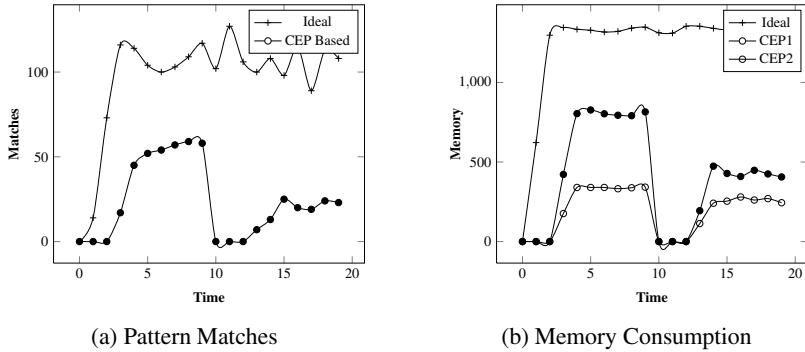
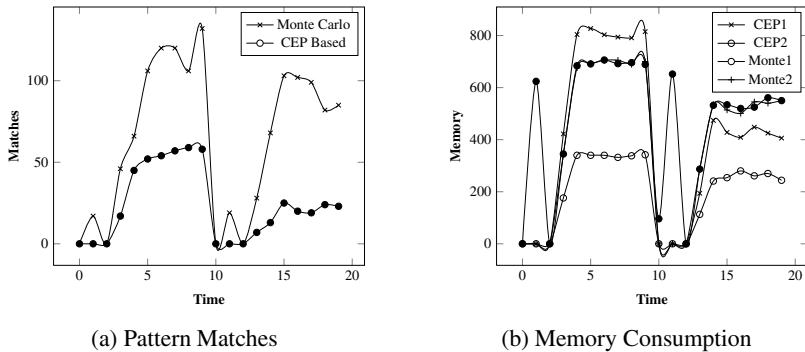


Fig. 5 Ideal Engine vs Monte Carlo

**Fig. 6** Ideal Engine vs CEP based Partitioning Algorithm**Fig. 7** Monte Carlo vs CEP based Partitioning Algorithm

overall memory characteristics because it selects smaller event spaces to consider, and it also uses coarse grained events to decide where to put fine grained engines. This operation doesn't consume much memory. The Monte Carlo, on the other hand, has to work as a centralized and memory overloaded engine.

4 Conclusion and Future Work

In this paper, we have introduced a simplified view of Complex Event Processing called *peer model*. Peer model aims to simplify the way of looking at CEP to easily introduce conceptual scaling optimizations. With this model, it is easier to think about distributed event collection and processing. We have also described practical mappings of this model in real world situations. Further, we have identified *the event space correlation problem* that stops current CEP models from scaling conceptually. We have introduced two partitioning algorithms and evaluated their performances in a simulated environment. The proposed partitioning has a semantic nature in a sense, that algorithms consider inner relations among events to uncover possible

relations between producers. Experimental results show that the Monte Carlo partitioning algorithm performs better with regards to matching capabilities, but suffers from memory consumption. The CEP based algorithm provides an overall better memory performance and is easier to deploy into high volume environments, but has significantly lower matching capabilities.

In the future, we plan to extend the peer model from a query semantics perspective. We will introduce distributed algorithms that will allow the deployment of CEP rules into the peer model and also allow for the collection of matching results from it. Additional studies of partitioning algorithms are also needed. We hope that the CEP based algorithm may be improved to give better matching performance. All of these efforts are aimed to create an Open Source platform Peer CEP that is written in the Java Programming language. The simulator which was used to conduct experiments for this paper will also be part of the suite - for testing and research needs.

References

1. Jacobs, A.: The pathologies of big data. *Communications of the ACM - A Blind Person's Interaction with Technology CACM Homepage Archive* 52(8), 36–44 (2009)
2. Van Renesse, R., Birman, K.P., Vogels, W.: Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Trans. Comput. Syst.* 21(2) (2003)
3. Luckham, D.: *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*, 376 p. Addison-Wesley Professional, New York (2002)
4. Tovarňák, D.: Towards Multi-Tenant and Interoperable Monitoring of Virtual Machines in Cloud. In: *14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, Timisoara, Romania (2012)
5. Nguyen, F., Pitner, T.: Information System Monitoring and Notifications Using Complex Event Processing. In: *Proceedings of the Fifth Balkan Conference in Informatics*, Novi Sad, pp. 211–216. ACM, Serbia (2012) ISBN 978-1-4503-1240-0
6. Artikis, A., Etzion, O., Feldman, Z., Fournier, F.: Event processing under uncertainty. In: *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems (DEBS 2012)*, pp. 32–43. ACM, New York (2012)
7. Isoyama, K., Kobayashi, Y., Sato, T., Kida, K., Yoshida, M., Tagoto, H.: A scalable complex event processing system and evaluations of its performance. In: *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, DEBS 2012*. ACM, New York (2012)
8. Lee, S., Lee, Y., Kim, B., Candan, K.S., Rhee, Y., Song, J.: High-performance composite event monitoring system supporting large numbers of queries and sources. In: *Proceedings of the 5th ACM International Conference on Distributed Event-based System, DEBS 2011*, pp. 137–148. ACM, New York (2011)
9. Akram, S., Marazakis, M., Bilas, A.: Understanding and improving the cost of scaling distributed event processing. In: *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, DEBS 2012*, pp. 290–301. ACM, New York (2012)
10. Wu, E., Diao, Y., Rizvi, S.: High-performance complex event processing over streams. In: *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, SIGMOD 2006*, pp. 407–418. ACM, New York (2006)

11. Randika, H.C., Martin, H.E., Sampath, D.M.R.R., Metihakwala, D.S., Sarveswaren, K., Wijekoon, M.: Scalable fault tolerant architecture for complex event processing systems. In: International Conference on Advances in ICT for Emerging Regions (ICTer), Colombo, Sri Lanka (2010)
12. Biger, A., Etzion, O., Rabinovich, Y.: Stratified implementation of event processing network. In: 2nd International Conference on Distributed Event-Based Systems. Fast Abstract, Rome, Italy (2008)
13. Minařík, P., Vykopal, J., Krmíček, V.: Improving Host Profiling with Bidirectional Flows. In: International Conference on Computational Science and Engineering, CSE 2009, August 29-31, vol. 3, pp. 231–237 (2009)
14. Kaarela, P., Varjola, M., Noldus, L.P.J.J., Artikis, A.: PRONTO: support for real-time decision making. In: Proceedings of the 5th ACM International Conference on Distributed Event-Based System, DEBS 2011, pp. 11–14. ACM, New York (2011)

A Heuristic to Explore Trust Networks Dynamics

Vincenza Carchiolo, Alessandro Longheu,
Michele Malgeri, and Giuseppe Mangioni

Abstract. The shifting of computation and storage towards distributed, embedded and mobile systems encompasses several advantages, as the increasing in performance, scalability, and reduction of costs. A significant drawback though is the potential lack in security and trust, due to a huge number of different, dispersed devices. In such a context, trust-based networks allow to rank nodes (e.g. devices) so that only the *best* are chosen to store and manipulate users personal data. However, the study of rank assessment and how a node can improve its value may become a hard task, especially for large networks. Therefore, in this work we propose a heuristic based on statistical considerations that allows a node to achieve a good rank via a proper selection of trust links with other nodes, preserving both the effectiveness and the efficiency.

1 Introduction

The shifting of computation and storage as well as of many social interactions towards the world of distributed, embedded and mobile systems recently led to neologisms such as *cloud computing*, *ubiquitous computing*, *everyware* and many others [1] [2] [3].

These phenomena encompass several advantages, as the improvement in users interaction, performance, scalability, availability and the reduction of costs. A significant drawback though is the potential lack in data security and confidentiality, due to a huge number of different worldwide dispersed devices, so that end users no longer perceive who/what and where their personal data are stored [4] [5] [6]. In such a context, trust-based networks can be effectively used to guarantee the

Vincenza Carchiolo · Alessandro Longheu ·
Michele Malgeri · Giuseppe Mangioni
Dip. Ingegneria Elettrica, Elettronica e Informatica,
Facoltà di Ingegneria, Università degli Studi di Catania, Italy
e-mail: car@diit.unict.it,
{alessandro.longheu,michele.malgeri,
gmangioni}@dieei.unict.it

reliability of unknown actors (people, devices, resources, services, hardware infrastructures etc.), in particular trust values allow to rank actors (nodes of the network) so that only the *best* are considered.

While the study of trust (and rank) assessment has been addressed even in large networks [7] [8] [9], the dynamics of trust networks has been only partially considered within existing metrics; note that here the term *dynamics* is intended as the changing in ranks classification caused by the process of joining to (or leaving) the network (others adopt a different meaning, e.g. [10] considers the dynamics as the evolution of trust over time due to the change in nodes behavior).

In this scenario, whenever a new node joins the network it establishes a certain number of trust links with other existing nodes in order to get trusted, aiming at improving its rank. Note however that in real systems each time an attachment to a node occurs, this involves some cost for the new node, thus we should discover which (and how many) trust links should be established to provide a good rank at a low cost. Such a multi objectives optimization problem can be hard to solve, especially when large networks are considered (as frequently happens in real scenarios); to get an answer without performing exhaustive experiments with all possible links combinations, in this paper we propose a heuristic based on statistical considerations aiming at exploiting just *relevant* links combinations, preserving both the effectiveness (rank enhancement) and the efficiency (less expensive as possible).

The paper is organized as follows. In section 2 we outline the model of trust network we are focusing on, together with all definitions needed to outline the scenario whereas in section 3 we propose a heuristic to avoid the complexity of considering exhaustive approach; in section 4 we show simulations used to validate the proposed heuristic, finally providing concluding remarks and future works in section 5.

2 The Trust Network: Model and Definitions

As outlined in the introduction, our aim is to evaluate an efficient strategy in the attachment process a new node is engaged with, addressing the goal of getting a good rank while limiting the effort.

The scenario we consider is that of a trust network, modeled as a graph where the nodes (N) are agents (persons, devices, resources etc) and the arcs (E) represent trusting relationships, each labeled with a measure (\mathcal{L}) of the trust value according to a given metric. This model is widely accepted in literature [11, 12, 13] and largely inspired by the PageRank [14] algorithm that allows us to assess a steady global trust value associated to each node.

Although the trustworthiness per se is important to study the evolution of the network, the key role here is played by the *rank*, i.e. the placement each node achieves with respect to others. We point out that a node's rank is proportional to its trustworthiness, i.e. the more trusted the node, the better its rank. Here we simply defined the rank as the position a node gets when they are ordered in a descending way according to their trust values; other analytical trust-vs-rank relationships could be considered, however this is out of our scope.

Moreover, in the following we do not impose any specific metric for trust. Among many proposals that exist in literature [7, 8, 9, 10, 15], we used EigenTrust [7] in our experiments since it is one of the simplest yet efficient metric in evaluating global trust; note however that any other metric can be adopted since our proposal does not rely on specific metric features.

A further hypothesis is that the agent joining the trust network is supposed to be *honest*, so it simply aims at gaining the best rank and has no other goals, as for instance using its (good) trust to subvert others' trust values; if so, a limit to the trust achievable by a node must be established, though we are not addressing such cases here.

To complete the scenario overview, we also have to formalize the attachment *effort* cited in the introduction, indeed in real networks an existing node does not trust new ones unless it assumes some responsibility and/or offers some service, for instance in P2P networks a peer must guarantee a given bandwidth or a minimum number of replica files, or in a social network a person must somehow sacrifice his/her privacy to be known (hence, trusted) by others. We then define the *effort* as the cost the agent X bears to persuade another agent Y to trust it; of course, if X aims at being trusted by more nodes it has to spend more, therefore the effort of X is:

$$\text{effort}_X = \sum_j c_{jX} \quad (1)$$

where c_{jX} are the normalized local trust values X receives from his neighbors j according to the EigenTrust metric we chose.

Simply put, the effort measures how many trust links the new agent must collect from others until it obtains a given trust; note that using values provided by EigenTrust does not affect this definition, i.e. any other metric can be used as well.

Finally, for the sake of simplicity, before the new agent X joins the network, all existing arcs are labeled with 1.0 as trust value. We choose this setting in order to avoid the distribution of trust values affects our simulation results. Note that the 1.0 trust value just refers to each *direct* trust, the global trust of each agent is instead evaluated with the EigenTrust metric and falls in the range [0,1];

As soon as X joins the network, some questions arise: (1) which (existing) nodes should X connect to in order to get a good rank since the beginning? (2) should X retains previously established links when adding new ones or a better rank could be achieved by also replacing some of existing trust links? In a few words, which is the best set of trust links that allows a node to achieve the best rank?

The idea to investigate on a non-random network attachment is also present in other works, e.g. [16], where focusing on social networks leads to consider the assortativity [17] and the clustering effect [18] as properties that determine a non-random network's dynamic behavior. The study we perform here is conducted on both random and scale-free networks, and we do not assume any hypothesis on the type of network (e.g. social, biological, etc.).

The simplest approach to study how trust values affect ranks evolution is a *brute force* algorithm, where we attempts to consider all configurations and compare the results [19]. However, this approach is too complex for (even not so) large size

networks: given a node X attempting to join a network with N nodes, assuming that local trust-values are $\{0, 1\}$, the available configurations to analyze are $2^N - 1$. Being this a value that easily explodes even for little networks, we need a criterion for discarding those configurations that will not lead to high rank. In the next section we present our heuristic approach based on some statistical considerations.

3 The Heuristic for the Reduction of the Effort-Rank Space

In order to study a strategy that allows a joining node to gain the best rank, we are interested in exploring the effort-rank space. For a network with N nodes this means to compute the rank for all links combinations a joining node can have with the N nodes of the network; since such number is $2^N - 1$, this makes practically unfeasible the study of many real networks. For instance, the network used in our experiments has $N = 20$ nodes meaning that a brute force algorithm has to take into account 1 048 575 configurations. This approach permits to study networks whose dimension is up to 25-30 nodes (a network with 30 nodes requires 1 073 741 823 configurations to be analyzed), but real networks can be even bigger. To solve this problem, we want to reduce the configurations by discarding those not *relevant* via the heuristic approach explained in the following.

The driving idea is based on the fact that both rank and effort can range over a limited amount of values, therefore it is possible to collapse all configurations achieving a given rank into just one.

Each node can assume a trust-based rank bounded to an upper limit that linearly depends on the number N of nodes of a network: it ranges indeed from 1 to N (where 1 is the best and N the worst rank). Therefore, a node attempting to join a network whose size is N can gain a rank in the range $[1, N + 1]$, so we can study no more than $N + 1$ configurations. Moreover, also the $effort_X$ linearly depends on N and ranges from 1 to N in the hypothesis that local trust-values are in $\{0, 1\}$.

Thanks to the linear dependence of both rank and effort, we can simply find the upper bound of the number of points in the effort-rank space that is $N \cdot (N + 1)$, much lesser than $2^N - 1$. This can be explained by the fact that several configurations (corresponding to several trust-values) actually provide the same rank.

Since we are interested in the rank a node can assume, it would be interesting to find a way to directly explore this space. This leads to a considerable decrease in the number of configurations to compute. For instance, for a network with $N = 20$ nodes, a brute force algorithm has to take into account 1 048 575 configurations, whilst the number of rank points are 421 ($20 \cdot (20 + 1)$) at most, thanks to the fact that many configurations map to one rank, for example $effort_X = 10$ leads to the same rank 184 756 times.

Based on these considerations, the best approach (i.e. with less computational time cost) to explore the effort-rank space is to compute at most $N \cdot (N + 1)$ points. The problem is to select exactly $N \cdot (N + 1)$ attachment configurations among the possible $2^N - 1$ that guarantee the maximum coverage. In other words, we want to exclude those attachment configurations that produce the same points in the

effort-rank plane. To do this we need a heuristic that permits to select ideally just one attachment configuration per point.

The heuristic we propose in this work is based on the analysis of the statistical distributions of the attachment configurations in the effort-rank plane. To introduce it we need some notations.

We call in^X the in-set of a node X (i.e. the set of nodes for which an outgoing link to the node X exists):

$$in^X = \{j : (j, X) \in E\} \quad (2)$$

We also refer to this set as the *attachment configuration* of the node X . Note that the cardinality of in^X is equal to the $effort_X$ just in the case the direct local trust values are $\{0, 1\}$.

A new node joining the network can be linked to several different nodes, thus originating several different attachment configurations. Let us suppose to enumerate such configurations by using a subscript index. So, let us consider for a given effort two attachment configurations of the node X , say in_p^X and in_q^X ; we define their distance as:

$$d(in_p^X, in_q^X) = |in_p^X - in_q^X| \quad (3)$$

i.e. the cardinality of the difference between the two in-sets of the node X . Note that the minimum distance d is 1, while the maximum value is given by:

$$\text{maximum}\{d(in_p^X, in_q^X)\} = \min\{|in_p^X|, N - |in_p^X|\} \quad (4)$$

To find a good heuristic in exploring the effort-rank space, we examined the statistical distribution of the distance, considering all the configurations having the same effort. In table 1 we show some of the $2^N - 1$ probabilities that two configurations with the maximum distance either belong (P_{in}^{maxd}) or not (P_{out}^{maxd}) to the same point in the effort-rank space. In other terms, given an attachment configuration A , the table reports the chance that another configuration B whose distance from A is the maximum (as defined by eq. 4) leads to a different rank value (different point in the space). The table 1 reports only the first points, i.e. with effort 1, 2 and 3.

Except for the case with effort 1 and rank 20 where only one point is present, the table 1 highlights that the configurations whose distance is maximum have a high probability to get a different point of the plane. This distribution suggest us a strategy to generate attachment configuration: for a given effort, the configuration must be chosen in such a way that the distance between any two of them is maximum.

In summary, the heuristic we propose is to start with a given configuration and, for each value of the effort, generate a sequence of other attachment configurations whose distance from the previous one is the maximum (as given by eq. 4). Then we apply again the previous step by starting from a new attachment configuration just in case we have not found all the $N + 1$ points of the plane. This approach allows to drastically decrease the number of attachment configurations to analyze as shown in details in the next section.

Table 1 Probability to stay in / to escape from a given point of the effort/trust-rank plane

Effort	Rank	P_{in}^{maxd}	P_{out}^{maxd}
1	20	1	0
2	20	0.3312	0.6688
2	19	0.5337	0.4663
2	18	0.0514	0.9486
3	19	0.0082	0.9918
3	18	0.0159	0.9841
3	17	0.1258	0.8742
3	16	0.2041	0.7959
3	15	0.1978	0.8022
3	14	0.1448	0.8552
3	13	0.1151	0.8849
3	12	0.0710	0.9290
3	11	0.0507	0.9493
3	10	0.0146	0.9854
3	9	0.0088	0.9912
3	8	0.0015	0.9985
...

4 Simulation Results

The heuristic discussed in the previous section facilitate the study of dynamics and behavior of large networks, but we still have to validate our approach by comparing the results provided by the brute force algorithm with those coming from the proposed heuristic. Given the cited computational limit concerning the size of network, we compare such results for networks with a size of 20 nodes.

To avoid the introduction of biasing we synthesized several networks with different distributions (random, scale-free) and created several ad-hoc topologies (we call them *regular*) that preserve the generality of outcoming networks. Each group of networks has been studied using both the proposed heuristic and brute force algorithm.

The figure 1 reports the effort-rank graph for a network of 20 nodes analyzed with the brute-force algorithm, therefore all the possible attachment configurations are presents. Each point in the graph may represent several different configurations

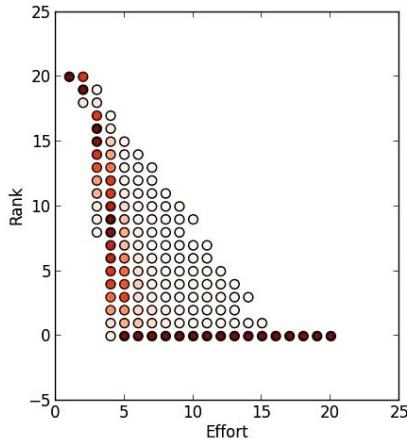


Fig. 1 Regular Net - Brute force analysis

(some points result from more than 156 000 input configurations) raising the complexity and time of the simulations. To qualitatively distinguish such points, different colors are used to indicate the *density*, in particular the more input configurations map to the same point, the darker that point is represented in figure.

The figure 2 reports the same graph of figure 1 applied to the same networks but using the proposed heuristic. The figures highlight that results are a good approximation since almost all points (120 over 160) have been found, showing the same pattern (dynamics) and limits; the points that have been discarded are not particularly meaningful for the attachment strategy and the rank distribution (being the pattern preserved from fig. 1 to fig. 4). Also note that in columns where the heuristic approach does not capture all points, e.g. that for $effort = 3$, the most significant points to define the correct range of rank are still detected. The outcome of capturing the correct range is very important to define the correct attachment strategy, i.e. the points to connect to obtain the best rank starting from current configuration. We believe this is the most common case since in real world a node is not likely to drop (possibly recent) connections established with other nodes.

Of course the density is not preserved by the heuristic approach due to heuristic itself that tries to find as less configurations as possible for each point in the effort-rank space.

We also compared the brute force algorithm with the heuristic on synthesized random and scale-free networks, both with a dimension of 20 nodes. The figures 3 and 4 show the results, that are similar for all networks we analyzed, thus confirming that the heuristic approach captures both the general dynamics and most of the significant points to define a step-by-step attachment strategy aiming at obtaining that best rank with less effort.

The time complexity we obtained is about 90% of the computational time thanks to the reduction of the configurations to consider: about 10 000 instead of the 1 048 575.

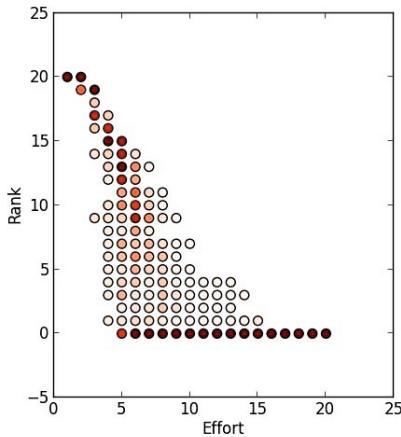


Fig. 2 Regular Net - Heuristic analysis

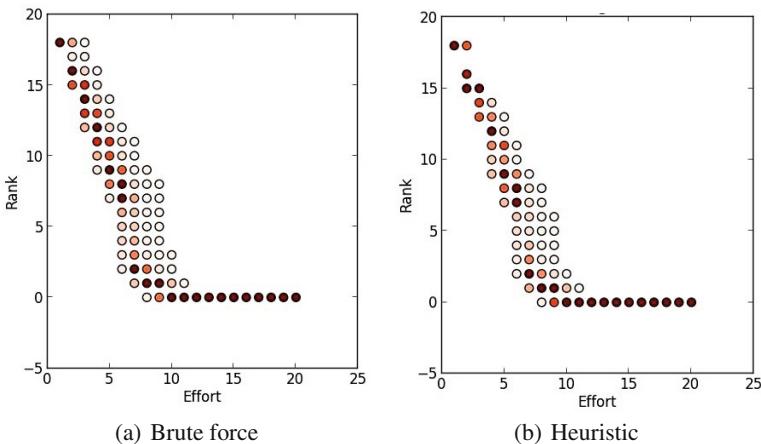


Fig. 3 Random networks simulations

5 Final Remarks

The use of ranking and trust networks is becoming more and more relevant in many contexts. Real cases however exhibit large scale, i.e. networks with many nodes, so a strategy for assessing nodes rank in an efficient and effective way is required. Here we presented a heuristic to make it feasible the study of which nodes a new one should attach to in order to increase its rank with a minimal effort. We compared such heuristic versus a brute force algorithm for three network types: random, scale free and a *regular* network. We showed both the effectiveness of the approach,

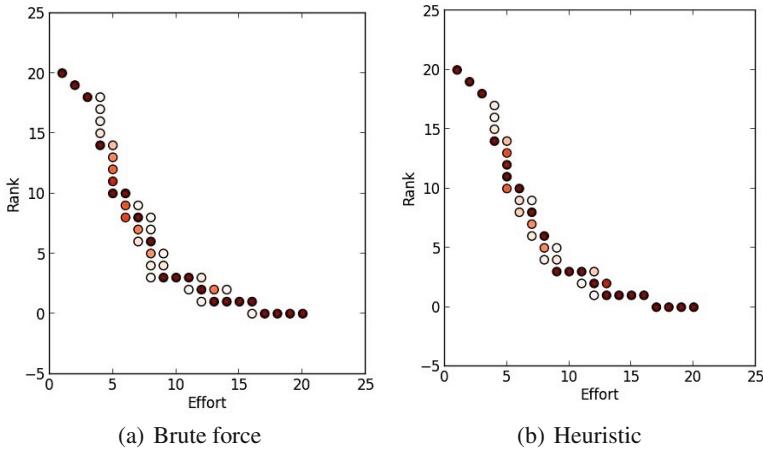


Fig. 4 Scale Free networks simulations

thanks to the dynamics and ranks range preservation, and the efficiency, being the simulation time up to 90% shorter than brute force. Further works include:

- the simulation of the proposed heuristic on larger networks, for which the direct comparison with brute force will be unfeasible, but results would be more significant;
- the investigation on possibly more effective heuristics, in order to improve the reduction of the effort-rank space, especially for large networks, as well as the comparison with a formal, analytical approach (i.e. multi-objective optimization)
- the analysis of other trust metrics and different nodes behavior (e.g. whitewashing, group empowering etc.)
- the study of potential application of such heuristics in real cases.

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. *Commun. ACM* 53(4), 50–58 (2010)
2. Hansmann, U., Nicklous, M.S., Stober, T.: *Pervasive computing handbook*. Springer-Verlag New York, Inc., New York (2001)
3. Nieuwdorp, E.: The pervasive discourse: an analysis. *Comput. Entertain.* 5(2), 13 (2007)
4. Santos, N., Gummadi, K.P., Rodrigues, R.: Towards trusted cloud computing. In: Proceedings of the 2009 Conference on Hot Topics in Cloud Computing, HotCloud 2009. USENIX Association, Berkeley (2009)
5. Dietrich, K., Winter, J.: Implementation aspects of mobile and embedded trusted computing. In: Chen, L., Mitchell, C.J., Martin, A. (eds.) *Trust 2009*. LNCS, vol. 5471, pp. 29–44. Springer, Heidelberg (2009)

6. Wang, Y., Norice, G., Cranor, L.F.: Who is concerned about what? a study of american, chinese and indian users' privacy concerns on social network sites. In: McCune, J.M., Balacheff, B., Perrig, A., Sadeghi, A.-R., Sasse, A., Beres, Y. (eds.) Trust 2011. LNCS, vol. 6740, pp. 146–153. Springer, Heidelberg (2011)
7. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The eigentrust algorithm for reputation management in P2P networks. In: Proceedings of the Twelfth International World Wide Web Conference (2003)
8. Zhou, R., Hwang, K., Cai, M.: Gossiptrust for fast reputation aggregation in peer-to-peer networks. *IEEE Trans. on Knowl. and Data Eng.* 20(9), 1282–1295 (2008)
9. Zhou, R., Hwang, K.: Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing. *IEEE Trans. Parallel Distrib. Syst.* 18(4), 460–473 (2007)
10. Walter, F.E., Battiston, S., Schweitzer, F.: Personalised and dynamic trust in social networks. In: Bergman, L.D., Tuzhilin, A., Burke, R.D., Felfernig, A., Schmidt-Thieme, L. (eds.) RecSys, pp. 197–204. ACM (2009)
11. Marsh, S.: Formalising trust as a computational concept. Technical report, University of Stirling. PhD thesis (1994)
12. Golbeck, J.A.: Computing and applying trust in web-based social networks. PhD thesis, College Park, MD, USA, Chair-Hendler, James (2005)
13. Walter, F.E., Battiston, S., Schweitzer, F.: A model of a trust-based recommendation system on a social network. *Journal of Autonomous Agents and Multi-Agent Systems* 16, 57 (2008)
14. Berkhin, P.: A survey on pagerank computing. *Internet Mathematics* 2(1), 73–120 (2005)
15. Xiong, L., Liu, L.: Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Trans. Knowl. Data Eng.* 16(7), 843–857 (2004)
16. Allodi, L., Chiodi, L., Cremonini, M.: Modifying trust dynamics through cooperation and defection in evolving social networks. In: McCune, J.M., Balacheff, B., Perrig, A., Sadeghi, A.-R., Sasse, A., Beres, Y. (eds.) Trust 2011. LNCS, vol. 6740, pp. 131–145. Springer, Heidelberg (2011)
17. Newman, M.E.: Assortative Mixing in Networks. *Physical Review Letters* 89(20), 208701 (2002)
18. Opsahl, T., Panzarasa, P.: Clustering in weighted networks. *Social Networks* 31(2), 155–163 (2009)
19. Carchiolo, V., Longheu, A., Malgeri, M., Mangioni, G.: Gain the best reputation in trust networks. In: Brazier, F.M.T., Nieuwenhuis, K., Pavlin, G., Warnier, M., Badica, C. (eds.) Intelligent Distributed Computing V. SCI, vol. 382, pp. 213–218. Springer, Heidelberg (2011)

Resource Scaling Performance for Cache Intensive Algorithms in Windows Azure

Marjan Gusev and Sasko Ristov

Abstract. Customers usually expect linear performance increase for increased demand for renting resources from cloud. However, it is not always the case, although the cloud service provider offers the specified infrastructure. The real expectation is limited due to memory access type, how data fits in available cache, nature of programs (if they are computation-intensive, memory or cache demanding and intensive) etc. Cloud infrastructure in addition rises new challenges by offered resources via virtual machines. The ongoing open question is choosing what is better - usage of parallelization with more resources or spreading the job among several instances of virtual machines with less resources. In this paper we analyze behavior of Microsoft Windows Azure Cloud on different loads. We find the best way to scale the resources to speedup the calculations and obtain best performance for cache intensive algorithms.

Keywords: Cloud Computing, HPC, Matrix Multiplication.

1 Introduction

Customers usually think about cloud computing as infinite pool of resources that offers an availability to rent resources on-the-way as much as they like. Cloud service providers (CSPs) offer computing and storage capacity by infrastructure organized in virtual machine (VM) instances. Scientists can collaborate with each other sharing the data in the cloud [1].

Amazon, Microsoft and Google offer on demand VM instances with 1, 2, 4 and 8 CPUs. The renting model is linear with constant price / performance ratio and the

Marjan Gusev · Sasko Ristov
Ss. Cyril and Methodious University,
Faculty of Information Sciences and Computer Engineering,
Rugjer Boshkovikj 16, 1000 Skopje, Macedonia
e-mail: {marjan.gushev, sashko.ristov}@finki.ukim.mk

price per hour doubles for doubled resources [2, 3, 11]. However, the performance indicators mentioned by CSPs are not sufficient to determine the actual performance of a virtual machine instance [9].

Speedup as performance measure expresses the relative ratio of time in the parallel environment in comparison to the sequential execution. Gustafson specifies the scaled speedup bounded to the linear speedup [6]. However, superlinear speedup (greater than the number of processors) is achieved in a single-tenant multi-thread environment in Windows Azure [5]. In this paper we continue the research about superlinear speedup in a multi-tenant single-thread and multi-thread environments on Windows Azure.

Cache intensive algorithms are those where the average number of accesses per element (reuse of element) is greater than 1 [14]. We use matrix multiplication algorithm (MMA) cache intensive algorithm with $O(N)$ computational complexity. Windows Azure is the platform where we make the experiments and make conclusions how to achieve maximum performance for cache intensive algorithms using the same number of CPU cores and paying the same price. The rest of the paper is organized as follows: Section 2 presents related works on performance in different cloud environments. The testing methodology along with description of environment and test cases is presented in Section 3. The results of the experiments about performance are elaborated in Section 4. The infrastructure impact to MMA performance is presented and analyzed in sections 5.1 and 5.2. Conclusion and future work are discussed in Section 6.

2 Related Work

Several papers refer to different performance behavior when scaling the resources in the cloud. Lu et al. discovered several pitfalls resulting in waste of active VMs (idle usage) [10]. They examine several pitfalls in Windows Azure Cloud during several days of performing the experiments: Instance physical failure, Storage exception, System update. Another conclusion is that Windows Azure does not work well for tightly-coupled applications [15].

Latest results show that the performance for a particular algorithm depends not only on the available CPU and RAM memory, but also on I/O, storage capacity, CPU cache architecture, communication latency runtime environment, platform environment etc [13].

Virtualization as a technique also impacts the performance. Less cache misses for cache intensive algorithms are reported in [4] in both single-tenant and multi-tenant resource allocation leading to better performance. Concurrent threads in homogeneous cloud multi-tenant environment do not finish simultaneously [13]. This is emphasized when the number of concurrent VM instances increases. Koh et al. determined a phenomenon, i.e., the same VM does not achieve the same performance at different times among the other active VMs on the same hardware [8]. VM granularity significantly effects the workload's performance for small network workload [16]. Therefore, performance isolation is necessary in cloud multi-tenant

environment [17]. Iakymchuk [7] determined that underutilization of resources by adding more nodes can improve the performance implementing more parallelism, i.e., the performance of the same resources on the same physical host provide worse performance rather than the same amount of resources on several physical machines.

3 Testing Methodology

In this section we present the testing methodology to offer reliable results.

3.1 Testing Environment

Windows Azure [12] is used as a testing environment. For each test case we use the same platform environment with different resource allocation. Runtime environment consists of C# with .NET framework 4 and threads for parallelization.

Each VM uses AMD Opteron 4171 HE processor(s) with 6 cores, but maximum 4 of 6 cores are dedicated per VM instance. Each core possesses 64 KB L1 data and instruction caches dedicated per core, 512KB L2 dedicated per core. L3 cache with total 5 MB is shared per chip.

3.2 Test Cases

We use simplified version of dense MMA with square matrices of same sizes $C_{N \times N} = A_{N \times N} \cdot B_{N \times N}$. Both sequential and parallel execution of MMA are performed for all test cases. We realize the experiments in each test case by varying the matrix size to analyze performance behavior upon different VM resources, overload and variable cache storage requirements. Sequential test cases define one thread to multiply the whole matrices $A_{N \times N}$ and $B_{N \times N}$ while parallel test cases specify each thread to multiply a part of row matrix $A_{N \times N/c}$ and the whole matrix $B_{N \times N}$ where $c = 1, 2, 4, 8$ denotes the total number of parallel threads.

Different test cases are defined by scaling Windows Azure VMs from **very large scale** using 1 Extra Large (*XL*) VM with 8 CPU cores; **large scale** with 2 Large (*L*) VMs with 4 CPU cores per VM; **medium scale** using 4 Medium (*M*) VMs with 2 CPU cores per VM; **small scale** with 8 Small (*S*) VMs with 1 CPU core per VM. The following paragraphs explains details for each test case.

Very Large scale (Test Case 1): Fig. 1 a) depicts the test case 1 activating one *XL* VM with 8 cores. The MMA is executed as one process in VM with 8 parallel threads which run on one core. Each thread multiplies a row of matrix $A_{N \times N/8}$ and matrix $B_{N \times N}$.

Large Scale (Test Case 2): This test case specifies 2 concurrent *L* VMs. The MMA is executed concurrently with 4 parallel threads per process (VM) and one process per VM as shown in Fig. 1 b). Each thread is specifying quarter of operation scheduled for each process that multiplies half of the matrix $A_{N \times N}$ divided horizontally, i.e. a row matrix $A_{N \times N/2}$ and the whole matrix $B_{N \times N}$.

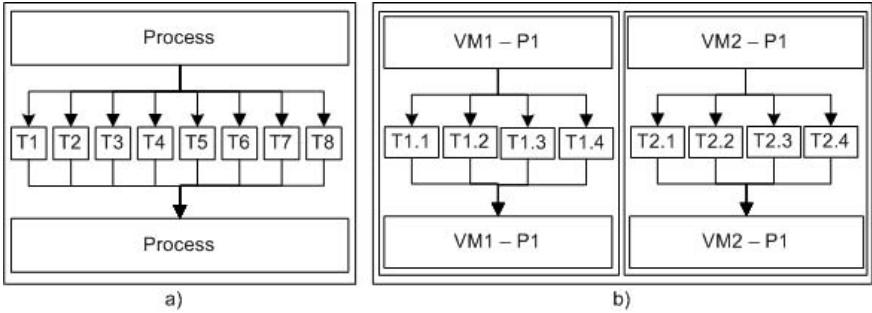


Fig. 1 Test Cases 1 (a) and 2 (b)

Medium Scale (Test Case 3): The medium scale test case activates 4 concurrent M VMs. One process with 2 threads is scheduled per VM to be executed on a different core as depicted in Fig. 2 a). MMA specifies multiplication of matrix $A_{N \cdot N/4}$ and the whole matrix $B_{N \cdot N}$ per process and the two threads per process perform half of these computations by multiplying $A_{N \cdot N/8}$ with matrix $B_{N \cdot N}$.

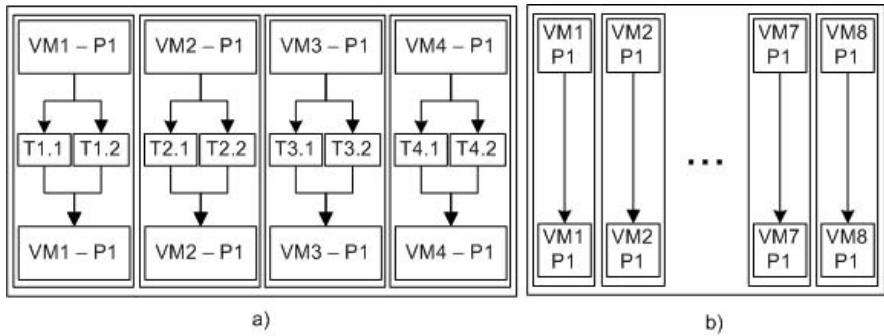


Fig. 2 Test Cases 3 (a) and 4 (b)

Small Scale (Test Case 4): In this test case there is straight forward programming using 1 thread per process, and 1 process per VM activating eight concurrent S VMs as shown in Fig. 2 b). Each process (and thread) executes MMA performing multiplication of matrix $A_{N \cdot N/8}$ with matrix $B_{N \cdot N}$.

Sequential Execution on Only One Core (Test Cases 5-8): In order to make comparison and analyze the speedup we define test cases 5-8 to execute MMA sequentially on already defined testing environments in previous 4 test cases. Each process executed per core performs all required multiplications of the whole matrices $A_{N \cdot N}$ and $B_{N \cdot N}$. Since only one core is used per test case the remaining 7 cores are unused.

3.3 Testing Goal

The experiments are realized with the following two goals:

- To determine the performance of a particular infrastructure by measuring the speedup in comparison to sequential execution; and
- To determine optimal hardware resource allocation in Windows Azure.

Computational *Speed V* is defined by (1) and *Speedup S* is calculated by (2), where indexes *Seq* and *Par* denote sequential and parallel execution times. In evaluation we use the average execution time *AET* per test case.

$$V = 2 \cdot N^3 / AET \quad (1)$$

$$S = ExecutionTime_{Seq} / AET_{Par} \quad (2)$$

Additionally we measure *relative speedup R_i* for sequential and parallel test cases. (3) defines the relative speedup of sequential execution in smaller VMs compared to the *XL*, i.e. test cases 6, 7 and 8 compared to test case 5. Using the same principle (4) defines the relative speedup of parallel execution in smaller VMs compared to the *XL*, i.e. test cases 2, 3 and 4 compared to test case 1. The index *i* denotes the corresponding test case.

$$R_{iSeq} = V_i / V_5 \quad (3)$$

$$R_{iPar} = V_i / V_1 \quad (4)$$

4 The Results of the Experiments

This section presents the results of the experiments that run test cases. A huge speed discrepancy that appears when more concurrent processes are executed [13]. Here we measure the average speed and speedup and analyze the dependencies obtained by different hardware resource allocation by comparing the results of test cases 1 and 5, 2 and 6, 3 and 7, and 4 and 8 as described in Section 3.2. Explanations also refer to regions *L₁*, *L₂*, *L₃* and *L₄* as regions where the complete data retired by the algorithm will fit in corresponding cache level.

The speedup achieved by comparing the test cases 1 to 5 for different matrix size *N* is presented in Fig. 3. We determine two main regions with different performance. For *N* < 572 (*L₃* region) the whole matrices can be placed in L3 cache and performance is much better than for *N* > 572 (*L₄* region) where L3 cache misses are generated. A superlinear speedup is determined in certain points of the *L₄* region, a phenomenon that happens due to doubled size L3 cache which is present for parallel execution in comparison to the sequential. More details on how and why superlinear speedup can be achieved in cloud virtual environment can be found in [14].

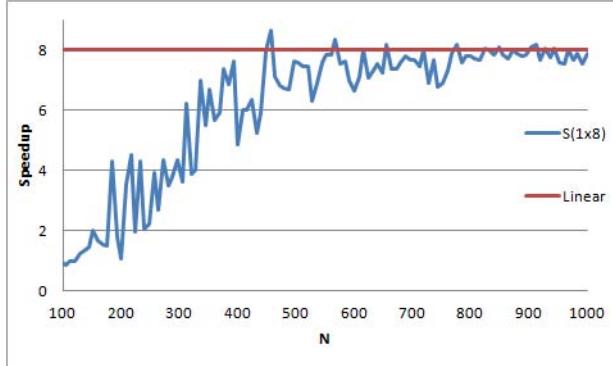


Fig. 3 Speedup comparing Test Cases 1 and 5

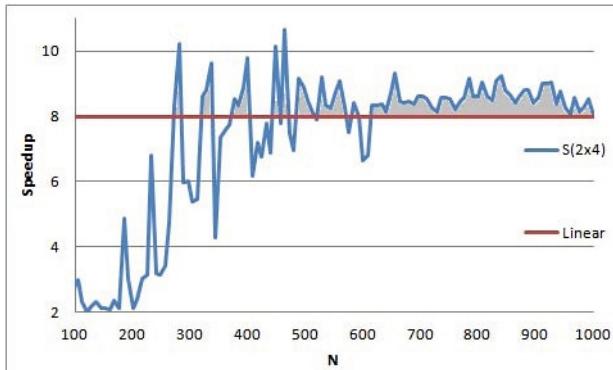


Fig. 4 Speedup comparing Test Cases 2 and 6

Fig. 4 depicts the speedup of test case 2 compared to test case 6 for different matrix size N . The same cache regions with different performance are identified in this test case also. We found that the whole L_4 region is a superlinear region since only half matrix A is stored in L3 cache for parallel execution rather than the whole matrix A for sequential execution.

Fig. 5 shows speedup resulted by comparison of test cases 3 and 7 for different matrix size N . We can conclude that the whole L_4 region is superlinear. This infrastructure provides even greater speedup than the test case 2.

Fig. 6 depicts the speedup comparing the test cases 4 to 8 for different matrix size N . The important result is the superlinear speedup in L_2 region since each VM has only one core which has dedicated L1 and L2 cache per core and in this case per VM. Entering the L_3 and L_4 region multi-tenancy provides more cache misses replacing the blocks that other VMs need from shared L3 cache. Therefore speedup in L_4 region is almost linear although we found superlinear speedup for some N .

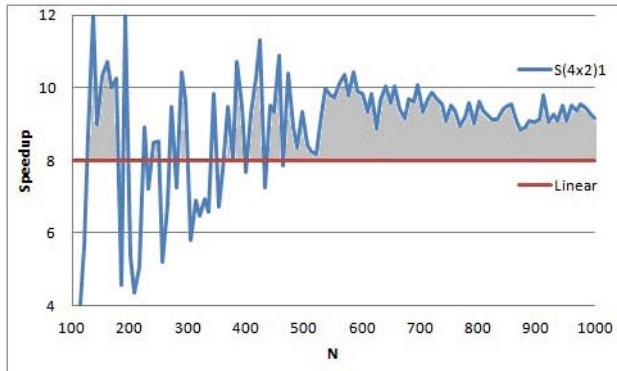


Fig. 5 Speedup comparing Test Cases 3 and 7

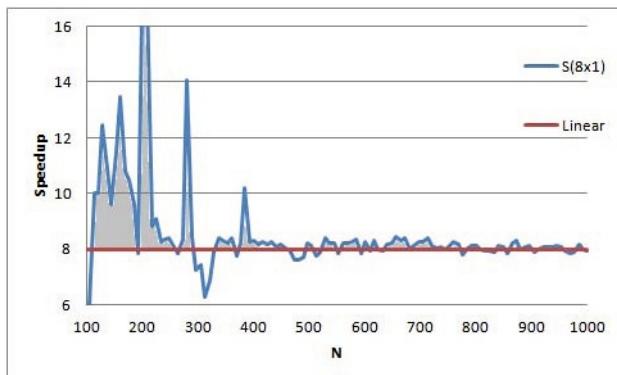


Fig. 6 Speedup comparing Test Cases 4 and 8

5 Which Orchestration Is Optimal?

This section describes the results of testing the performance impact of hardware infrastructure orchestration. We analyze the results to understand if single-tenant with multi-threading, or multi-tenant with multi-threading or multi-tenant with single-threading is the optimal environment to achieve maximum performance for MMA, expressed as faster execution and greater speedup.

5.1 Hardware Infrastructure Impact on Sequential Execution

Similar speed value is obtained in each test case and therefore we focus on relative speed depicted in Fig. 7. We can conclude that relative speeds are stable in L_2

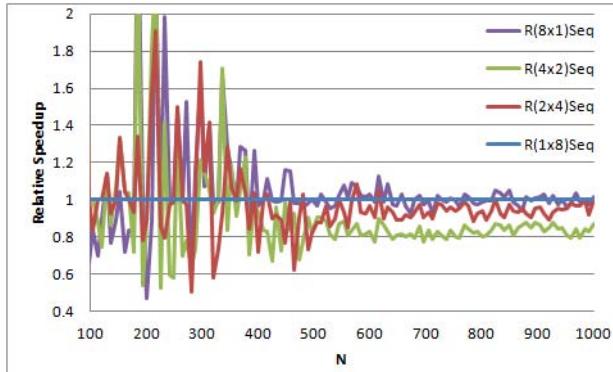


Fig. 7 Relative speedup R_{iSeq} for sequential executions

and L_4 regions rather than L_3 region. MMA algorithm best performance for sequential execution is on XL VM in front of Large, M and S for the L_2 region. However, XL and S VMs lead in front of L and M in L_4 region.

5.2 Hardware Infrastructure Impact on Parallel Execution

The performance of parallel execution is measured for test cases 1, 2, 3 and 4. The results for the speed and the relative speed are presented in Figure 8 for test cases 2, 3 and 4 compared to test case 1. We also observe the same three regions L_2 , L_3 and L_4 but with different results. The speed increases in L_2 region for the test cases with multi-threading, i.e. test cases 1, 2 and 3. The speed saturates in L_3 region and also in L_4 region with decreased value for all test cases.

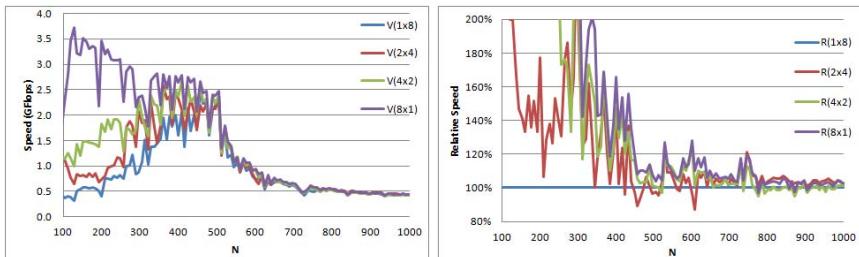


Fig. 8 Speed V (left) and Relative speedup R_{iPar} (right) for parallel executions on different environments

We can conclude that the best performance for parallel MMA is achieved using 8 *S* instances ahead of 4 *M* instances, 2 *L* or 1 *XL* for both *L*₂ and *L*₃ regions. The order is retained also in *L*₄ region where for huge matrices all environments provide similar performance and other algorithms should be used. Analyzing the performance behavior in figures 3, 4, 5 and 6 we can conclude that the environment defined by test case 4 is the leader in the speedup race in front of the test cases 3, 2 and 1 in *L*₂ region, and the environment for test case 3 is the leader for the speedup race in front of the test cases 2, 4 and 1 in regions *L*₃ and *L*₄.

6 Conclusion and Future Work

Dense MMA is computation-intensive, memory demanding and cache intensive algorithm that can be easily scaled to fit any hardware granularity. Therefore it is a good representative as test data algorithm for high performance computing. In this paper we have analyzed the performance of MMA using Windows Azure Cloud infrastructure. The experiments tested different cases using 1) single-tenant with multi-threading, 2) multi-tenant with multi-threading and 3) multi-tenant with single-threading environment hosted on the same hardware resources but with different spread and schedule among VMs.

The results of the experiments explaining the performance behavior for sequential executions confirmed the expectations that *XL* VM achieves maximum speed in front of Large, Medium and Small for *L*₂ region where the problem size fits in the *L*₂ cache. However we obtained that *S* VM achieves similar speed as *XL* VM and they lead in front of *L* and *M* VMs for the *L*₄ region, where the problem size is huge and needs extensive communication between cache and main memory.

This paper refers to unexpected (strange) results for parallel execution. MMA algorithm achieves maximum speed when executed parallel on 8 *S* instances, in front of 4 *M*, 2 *L*, and 1 *XL* for both the *L*₂ and *L*₃ regions, and almost all observed *L*₄ region. This means that the best performance can be achieved if MMA is granulated on 8 chunks and each chunk to be executed on 8 concurrent processes with one thread in *S* Windows Azure VM. The same environment achieves maximum speedup for the *L*₂ region. In *L*₃ and *L*₄ region maximum speedup is achieved if MMA is granulated on 4 chunks and each chunk to be executed on 4 concurrent processes with two threads in *M* VM.

Superlinear speedup achieved in a single-tenant multi-thread environment in Windows Azure is also present in a multi-tenant single-thread and other multi-thread environments. Even more, the speedup is greater in multi-tenant environment rather than single-tenant multi-thread environment. Greater speedup is achieved for granulated problem and executed using concurrent VMs instead of executing on one *XL* VM using parallelization.

We will continue with research on other hardware architectures and different clouds and exploit how MMA and other cache intensive algorithm executions depend on CPU and cache memory in order to find the optimal resource scheduling and scaling to obtain maximum performance.

References

1. Ahuja, S., Mani, S.: The State of High Performance Computing in the Cloud. *Journal of Emerging Trends in Computing and Information Sciences* 3(2), 262–266 (2012)
2. Amazon: EC2 (2013), <http://aws.amazon.com/ec2/>
3. Google: Compute Engine (2013), <http://cloud.google.com/pricing/>
4. Gusev, M., Ristov, S.: The Optimal Resource Allocation Among Virtual Machines in Cloud Computing. In: Proceedings of the 3rd International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2012), pp. 36–42 (2012)
5. Gusev, M., Ristov, S.: Superlinear Speedup in Windows Azure Cloud. In: 2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET, IEEE CloudNet 2012), Paris, France, pp. 173–175 (November 2012)
6. Gustafson, J., Montry, G., Benner, R.: Development of Parallel Methods for a 1024-processor Hypercube. *SIAM Journal on Scientific and Statistical Computing* 9(4), 532–533 (1988)
7. Iakymchuk, R., Napper, J., Bientinesi, P.: Improving High-performance Computations on Clouds Through Resource Underutilization. In: Proceedings of the 2011 ACM Symposium on Applied Computing, SAC 2011, pp. 119–126. ACM (2011)
8. Koh, Y., Knauerhase, R., Brett, P., Bowman, M., Wen, Z., Pu, C.: An Analysis of Performance Interference Effects in Virtual Environments. In: IEEE International Symposium on Performance Analysis of Systems Software, ISPASS 2007, pp. 200–209 (April 2007)
9. Lenk, A., Menzel, M., Lipsky, J., Tai, S., Offermann, P.: What Are You Paying For? Performance Benchmarking for Infrastructure-as-a-service Offerings. In: Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011, pp. 484–491. IEEE Computer Society, USA (2011)
10. Lu, W., Jackson, J., Ekanayake, J., Barga, R.S., Araujo, N.: Performing Large Science Experiments on Azure: Pitfalls and Solutions. In: CloudCom 2010, pp. 209–217 (2010)
11. Microsoft: Windows Azure (2013), <http://www.windowsazure.com/pricing/>
12. Padhy, R.P., Patra, M.R., Satapathy, S.C.: Windows Azure Paas Cloud: An Overview. *Int. J. of Comp. App.* 1, 109–123 (2012)
13. Ristov, S., Gusev, M., Osmanovic, S., Rahmani, K.: Optimal Resource Scaling for HPC in Windows Azure. In: Markovski, S., Gusev, M. (eds.) *ICT Innovations 2012. Web Proceedings*, Macedonia, pp. 1–8 (2012) ISSN 1857-7288, <http://www.ictinnovations.org/2012/>
14. Ristov, S., Kostoska, M., Gusev, M., Kirovski, K.: Virtualized Environments in Cloud can have Superlinear Speedup. In: Proceedings of the 5th Balkan Conference in Informatics, BCI 2012, pp. 8–13. ACM (2012)
15. Subramanian, V., Ma, H., Wang, L., Lee, E.J., Chen, P.: Rapid 3D Seismic Source Inversion Using Windows Azure and Amazon EC2. In: Proceedings of IEEE, SERVICES 2011, pp. 602–606. IEEE Computer Society (2011)
16. Wang, P., Huang, W., Varela, C.: Impact of Virtual Machine Granularity on Cloud Computing Workloads Performance. In: 2010 11th IEEE/ACM International Conference on Grid Computing, GRID, pp. 393–400 (October 2010)
17. Wang, W., Huang, X., Qin, X., Zhang, W., Wei, J., Zhong, H.: Application-Level CPU Consumption Estimation: Towards Performance Isolation of Multi-tenancy Web Applications. In: 2012 IEEE 5th International Conference on Cloud Computing, CLOUD, pp. 439–446 (June 2012)

Distributed Event-Driven Model for Intelligent Monitoring of Cloud Datacenters

Daniel Tovarňák, Filip Nguyen, and Tomáš Pitner

Abstract. When monitoring cloud infrastructure, the monitoring data related to a particular resource or entity are typically produced by multiple distributed producers spread across many individual computing nodes. In order to determine the state and behavior of a particular resource all the relevant data must be collected, processed, and evaluated without overloading the computing resources and flooding the network. Such a task is becoming harder with the ever growing volume, velocity, and variability of monitoring data produced by modern cloud datacenters. In this paper we propose a general distributed event-driven monitoring model enabling multiple simultaneous consumers a real-time collection, processing, and analysis of monitoring data related to the behavior and state of many distributed entities.

1 Introduction

With the emergence of distributed computing paradigms (e.g. grid) the importance of monitoring steadily grew over the past two decades and with the advent of *cloud computing* it rapidly continues to do so. When monitoring a distributed infrastructure such as grid or cloud, the monitoring data related to a particular entity/resource (e.g. message queue, Hadoop job, and database) are typically produced by multiple distributed producers spread across many individual computing nodes.

A two-thousand node Hadoop cluster (open-source implementation of MapReduce) configured for normal operation generates around 20 gigabytes of application-level monitoring data per hour [3]. However, there are reports of monitoring data rates up to 1 megabyte per second per node [4]. In order to determine the state and behavior of a resource all the relevant data must be collected, processed, and evaluated without overloading the computing resources and flooding the network.

Daniel Tovarňák · Filip Nguyen · Tomáš Pitner
Masaryk University, Faculty of Informatics
Botanická 68a, 60200 Brno, Czech Republic
e-mail: {xtovarn, xnguyen, tomp}@fi.muni.cz

In our research we are particularly interested in behavior monitoring, i.e. collection and analysis of data related to the actions and changes of state of the monitored resources (e.g. web service crash) as opposed to the monitoring of measurable state (e.g. disk usage). The goal of state monitoring is to determine if the *state* of some resource deviates from normal. Our goal, on the other hand, is to detect *behavior* deviations and their patterns.

The volume, velocity, and variability of behavior-related monitoring data (e.g. logs) produced by modern cloud datacenters multiply and there is a need for new approaches and improvements in monitoring architectures that generate, collect, and process the data. Also the lack of multi-tenant monitoring support and extremely limited access to provider-controlled monitoring information prohibits cloud customers to adequately determine the status of resources of their interest [12]. As the portfolio of monitoring applications widens the requirements for monitoring architecture capabilities grow accordingly. Many applications require huge amounts of monitoring data to be delivered in *real-time* in order to be used for intelligent *online* processing and evaluation.

The goal of this paper is to propose a novel distributed event-driven monitoring model that will enable multiple simultaneous consumers to collect, process, and analyze monitoring data related to the behavior and state of many distributed entities in real-time. The rest of this paper is organized as follows. In Section 2 we present basic terms and concepts used in this paper. Section 3 deals with the proposed distributed event-driven monitoring model. Section 4 concludes the paper.

2 Background

In this section we introduce basic terms, concepts, and principles that our model is founded on. Based on these principles we incrementally design basic cloud monitoring model that we build upon later.

We define **monitoring** as a *continuous and systematic collection, analysis, and evaluation of data related to the state and behavior of monitored entity*. Note that in the case of distributed computing environment (such as cloud) its state and behavior is determined by the state and behavior of its respective constituents (components). *State* of monitored entity is a measure of its behavior at a discrete point in time and it is represented by a set of state variables contained within a state vector [6]. *Behavior* is an action or an internal state change of the monitored entity represented by a corresponding event [6]. Computer logs (system logs, console logs, or simply *logs*) are widely recognized as one of the few mechanisms available for gaining visibility into the behavior of monitored resource [9] regardless if its operating system, web server or proprietary application. Therefore in our work we consider logs (log events) to be the primary source of behavior-related information.

To properly describe the respective phases of *monitoring process* we adhere to its revised definition originally introduced by Mansouri and Sloman in [7]. Terms used to denote entities participating in the process are based on the terminology presented

in Grid Monitoring Architecture [11] and later revised by Zanikolas and Sakellariou in [13]. The monitoring process is composed of the following stages: *Generation* of raw monitoring data by sensors; *Production*, i.e. exposure of the data via predefined interface; *Distribution* of the data from producer to consumer; its *Consumption* and *Processing*.

The production, distribution, and consumption stages are inherently related – the distribution depends on the fashion the monitoring data are produced, and consequently, the consumption is dependent on the way the data are distributed. Therefore, the three stages will be collectively referred to as **monitoring data collection**. Also, there is a difference between online and offline monitoring data processing and analysis. An online algorithm processes each input in turn without detailed knowledge of future inputs; in contrast an offline algorithm is given the entire sequence of inputs in advance [1]. Note that the use of online algorithms for processing and analysis do not necessarily (yet more likely) lead to real-time monitoring, and vice-versa, the use of offline processing algorithms do not necessarily prohibit it.

In our work we consider physical and virtual machines to be the primary producers of monitoring data via the means of their operating systems (without any difference between host and guest OS). For simplicity's sake we do not consider virtualization using bare-metal hypervisor.

Meng [8] observed that in general, monitoring can be realized in different ways in the terms of distribution of monitoring process across the participating components. *Centralized data collection and processing*, i.e. there is only one single consumer; *Selective data collection and processing*, i.e. the consumer wishes to consume only a subset of the produced data; *Distributed data collection and processing*, i.e. data are collected and processed in a fully decentralized manner.

In general, the communication and subsequent monitoring data transfer can be initiated both by consumer (i.e. *pull model*) and producer (i.e. *push model*) [10]. In the pull model the monitoring data are (usually periodically) requested by consumer from the producer. On the other hand, in the push model the data are transferred (pushed) to the consumer as soon as they are generated and ready to be sent (e.g. pre-processed and stored).

3 Distributed Event-Driven Monitoring Model

In this section we further extend the basic cloud monitoring model with multi-tenancy support whilst leveraging principles of Complex Event Processing to allow for real-time monitoring of large-scale cloud datacenters. The resulting model allows multiple simultaneous consumers to collect, process, and analyze events related to the behavior of many distributed entities. State-related events are also supported to allow for state-behavior correlations.

To achieve data reduction the data collection is primarily based on publish-subscribe interaction pattern for the consumers to specify which monitoring data they are interested in. More importantly, the subscription model allows for definition

of complex patterns and aggregations following a *pattern-based* publish-subscribe schema. To reasonably process a tremendous amounts of monitoring data generated by cloud datacenters distributed collection and processing is primarily considered. From the data evaluation point of view the goal is to allow consumers to collect events from many distributed sources, define complex subscriptions and consequently analyze and evaluate the incoming highly-aggregated events in their own way. The monitoring architecture following this model is intended to be the part of provider's cloud infrastructure.

Historically, traditional DBMSs (including *active databases*) oriented on *data sets* were not designed for rapid and continuous updates of individual data items required by online monitoring discussed in this paper and performed very poorly in such scenarios. As pointed out by Babcock et al. [2], to overcome these limitations a new class of data management applications emerged: Data Stream Management Systems (DSMSs) oriented on evaluating *continuous queries* over *data streams*. According to Babcock, data streams differ from the conventional relational data model in several ways: (1) the data elements in the stream arrive online; (2) the system has no control over the order in which data elements arrive to be processed, either within a data stream or across data streams; (3) data streams are potentially unbounded in size; (4) once an element from a data stream has been processed it is discarded or archived – it cannot be retrieved easily unless it is explicitly stored in memory, which is typically small relative to the size of the data streams.

Full-fledged DSMSs typically allow for quite expressive queries supporting many standard operations (e.g. averages, sums, counts) and also windows (e.g. sliding window, and pane window) to specify the particular portion of the incoming data stream. As pointed out in [5] whilst considerably capable, DSMSs are still focused on the traditional relational data and produce continuously updated query results (e.g. as output data stream). Detection of complex patterns of elements involving sequences and ordering relations is usually out of the scope of DSMSs.

Complex Event Processing (CEP) in general follows the same goals and principles as DSMSs, yet as it is apparent from the term, it is focused on the processing of a very specific type of data elements – events (event is an occurrence within a particular domain – computing infrastructure monitoring in our case).

In our previous work [12] we introduced the concept of event-driven producer of monitoring data using extensible schema-based format. We argued that unified representation of monitoring information in the form of events increases data correlation capabilities, makes processing easier, and avoid the complexity of monitoring architecture. Together with standard delivery channel it is an important step towards extensible and interoperable multi-cloud (inter-cloud) monitoring.

In our model, CEP can be perceived as an extension of pattern-based publish-subscribe schema enabling consumers to subscribe for complex (composite) events based on expressive queries, e.g. using sequence patterns, temporal constraints, windows, filters and aggregations. Typically, the complex events can be re-introduced to the data stream for further processing (i.e. creating new complex events) which we

consider to be very powerful. Example 1 represents an example of simple monitoring subscription in SQL-like declarative Event Processing Language used by Esper¹ CEP engine. Subscription S1 subscribes for complex events that can indicate possible password cracking attack using dictionary approach.

```
select hostname, username, success, count(*) as attempts
from LoginEvent.win:time(30 sec)
where attempts > 1000, success=false
group by hostname, username
```

Example 1. Subscription S1 using EPL

The flow of monitoring data (see Figure 1) in our model can be described as follows: The *sensors* generate raw monitoring data related to a specific entity (e.g. Hadoop job, SSH daemon, and CPU). Based on this data, *producers* register and instantiate *simple events* (i.e. an occurrence related to one or more *entities*) with clearly defined structure. *Consumers* then create *subscriptions* to instrument *processing agents* to perform one or more processing functions. Such a function takes stream of *events* as input and outputs a stream of *complex events*. When applicable, a single *subscription* is partitioned into several simpler *subscriptions* and distributed among several *processing agents* (and *producers* as well). The examples of common processing functions include: filtering, sequence detection, aggregation, and anomaly detection. The *consumers* then receive the *complex events* they previously subscribed for. In order to achieve *multi-tenancy* [12], *consumers* can be restricted to subscribe for *events* related to a particular entity.

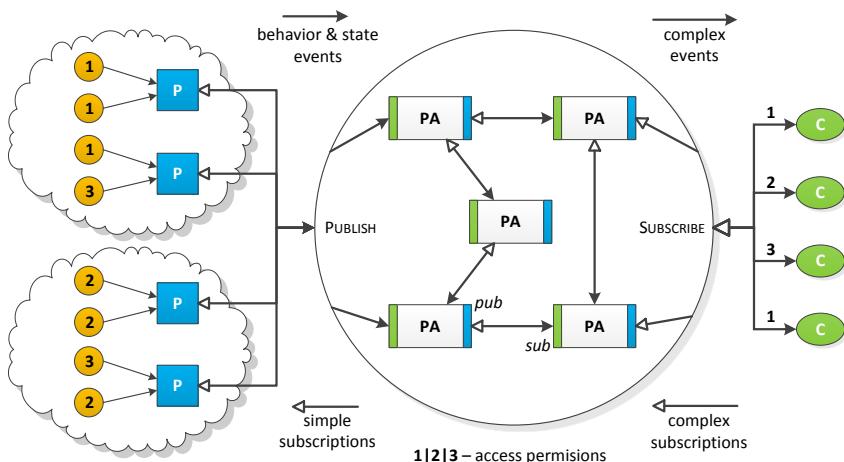


Fig. 1 Overview of Distributed Event-driven Monitoring Model in (Multi-)Cloud scenario

¹ <http://esper.codehaus.org/>

4 Conclusions and Future Work

In this paper we have presented a distributed event-driven model for intelligent cloud monitoring. An architecture following this model will allow multiple simultaneous consumers to collect, process, and analyze events related to the behavior of many distributed entities in real-time. We expect improvements in many areas when compared to traditional monitoring models based on offline algorithms. In future we plan to implement prototype of monitoring architecture following proposed model which will be then experimentally evaluated in the terms of intrusiveness, network overhead, and throughput with respect to the number of producers, consumers, volume, velocity, the variability of monitoring events, and the complexity and number of queries it is capable of dealing with. Multiple approaches for processing agents' topology, routing, query rewriting, and event distribution will be considered and evaluated.

References

1. Atallah, M.: Algorithms and Theory of Computation Handbook, 2 vol. set. CRC (1998)
2. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Principles of Database Systems. ACM, New York (2002)
3. Boulon, J., Konwinski, A., Qi, R., Rabkin, A., Yang, E., Yang, M.: Chukwa, a large-scale monitoring system. In: Proceedings of CCA (2008)
4. Crețu-Ciocârlie, G.F., Budiu, M., Goldszmidt, M.: Hunting for problems with artemis. In: Proceedings of the First USENIX Conference on Analysis of System Logs, WASL 2008, p. 2. USENIX Association, Berkeley (2008)
5. Cugola, G., Margara, A.: Processing flows of information: From data stream to complex event processing. ACM Comput. Surv. 44(3) (June 2012)
6. Mansouri-Samani, M.: Monitoring of distributed systems. PhD thesis, Imperial College London (University of London) (1995)
7. Mansouri-Samani, M., Sloman, M.: Monitoring distributed systems. IEEE Network 7(6) (November 1993)
8. Meng, S.: Monitoring-as-a-service in the cloud. PhD thesis, Georgia Institute of Technology (2012)
9. Oliner, A., Stearley, J.: What supercomputers say: A study of five system logs. In: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2007, pp. 575–584 (June 2007)
10. Rosenblum, D.S., Wolf, A.L.: A design framework for internet-scale event observation and notification. SIGSOFT Softw. Eng. Notes 22(6), 344–360 (1997)
11. Tierney, B., Aydt, R., Gunter, D., Smith, W., Swany, M.: A grid monitoring architecture. In: Global Grid Forum, pp. 1–13 (2002)
12. Tovariňák, D., Pitner, T.: Towards Multi-Tenant and Interoperable Monitoring of Virtual Machines in Cloud. In: SYNASC 2012, MICAS Workshop (September 2012)
13. Zanikolas, S., Sakellariou, R.: A taxonomy of grid monitoring systems. Future Generation Computer Systems 21(1), 163–188 (2005)

Programming Self-organizing Pervasive Applications with SAPERE

Franco Zambonelli, Gabriella Castelli, Marco Mamei, and Alberto Rosi

Abstract. SAPERE (“Self-aware Pervasive Service Ecosystems”) is a general framework to support the decentralized execution of self-organizing pervasive computing services. In this paper, we present the rationale underlying SAPERE and its reference conceptual architecture. Following, we sketch the middleware infrastructure of SAPERE and detail the interaction model implemented by it, based on a limited set of “eco-laws”. Finally, we show how in SAPERE one can express general-purpose distributed self-organizing schemes.

1 Introduction

Pervasive computing technologies are notably changing the ICT landscape, letting us envision the emergence of an integrated and dense infrastructure for the provisioning of innovative general-purpose digital services. The infrastructure will be used to ubiquitously access services for better interacting with the surrounding physical world and with the social activities occurring in it.

To support the vision, a great deal of research activity in pervasive computing has been devoted to solve problems associated to the development of effective pervasive service systems, including: supporting self-configuration and context-aware composition; enforcing self-adaptability and self-organization; and ensuring that service frameworks can be highly-flexible and long-lasting [12]. Unfortunately, most of the solutions so far proposed are in terms of “add-ons” to be integrated in existing frameworks [1]. The result is often an increased complexity of current frameworks and the emergence of contrasting trade-off between different solutions.

In our opinion, there is need for tackling the problem at the foundation, conceiving a radically new way of modeling integrated pervasive services and their

Franco Zambonelli · Gabriella Castelli · Marco Mamei · Alberto Rosi
Dipartimento di Scienze e Metodi dell’Ingegneria,
University of Modena and Reggio Emilia
e-mail: {franco.zambonelli,gabriella.castelli,marco.mamei,
alberto.rosi}@unimore.it

execution environments, such that the apparently diverse issues of context-awareness, dependability, openness, flexibility, can all be uniformly addressed once, and for all, via a sound and programmable self-organization approach. This is exactly the goal of SAPERE (www.sapere-project.eu), which proposes a novel nature-inspired approach to support the design and development of adaptive and self-organizing systems of pervasive computing services.

In this context, the contribution of this paper is twofold: *(i)* We present the overall conceptual architecture of the SAPERE approach, and show how it has been realized in the SAPERE middleware; *(ii)* We detail the specific approach to distributed self-organizing coordination promoted by SAPERE and discuss how this supports the effective development and execution of self-organizing pervasive applications.

2 The SAPERE Approach and Its Reference Architecture

SAPERE takes its primary inspiration from nature, and starts from the consideration that the dynamics and decentralization of future pervasive networks will make it suitable to model the overall world of services, data, and devices as a sort of distributed computational *ecosystem*.

As from Figure 1, SAPERE conceptually architects a pervasive service environment as a non-layered *spatial substrate*, laid above the actual pervasive network infrastructure. The substrate embeds the basic interaction laws (or *eco-laws*) that rule the activities of the system, and it represents the ground on which components of different species interact and combine with each other (in respect of the eco-laws and typically based on their spatial relationships), so as to serve their own individual needs as well as the sustainability of the overall ecology. Users can access the ecology in a decentralized way to use and consume data and services, and they can also act as “prosumers” by injecting new data or service components (possibly also for the sake of controlling the ecology behavior).

For the *components* living in the ecosystem, which we generically call “agents”, SAPERE adopts a common modeling and a common treatment. All agents in the ecosystem (and whether being sensors, actuators, services, users, data, or resources in general) have an associated semantic representation (in the case of pure data items, the entity and its representation will coincide), which is a basic ingredient for enabling dynamic unsupervised interactions between components. To account for the high dynamics of the scenario and for its need of continuous adaptation, SAPERE defines such annotations as living, active entities, tightly associated to the agent they describe, and capable of reflecting its current situation and context. Such *Live Semantic Annotations* (LSAs) thus act as observable interfaces of resources (similarly to service descriptions), but also as the basis for enforcing semantic and context-aware interactions (both for service aggregation/composition and for data/knowledge management).

The *eco-laws* define the basic interaction policies among the LSAs of the various agents of the ecology. In particular the idea is to enforce on a spatial basis, and

possibly relying on diffusive mechanisms, dynamic networking and composition of data and services. Data and services (as represented by their associated LSAs) will be sort of chemical reagents, and interactions and compositions will occur via chemical reactions, relying on semantic pattern-matching between LSAs. As it is detailed later on, the set of eco-laws includes: *Bonding*, which is the basic mechanism for local interactions between components, and acts as a sort of virtual chemical bond between two LSAs (i.e., their associated agents); *Spread*, which diffuses LSAs on a spatial basis, and is necessary to support propagation of information and interactions among remote agents; *Aggregate*, which enforces a sort of catalysis among LSAs, to support distributed data aggregation; *Decay*, which mimics chemical evaporation and is necessary to garbage collect data.

Adaptivity in the SAPERE approach will not be in the capability of individual components, but rather in the overall self-organizing dynamics of the ecosystem. In particular, adaptivity will be ensured by the fact that any change in the system (as well as any change in its components or in the context of the components, as reflected by dynamic changes in their LSAs) will reflect in the firing of new eco-laws, thus possibly leading to the establishment of new bonds or aggregations, and/or in the breaking of some existing bonds between components.

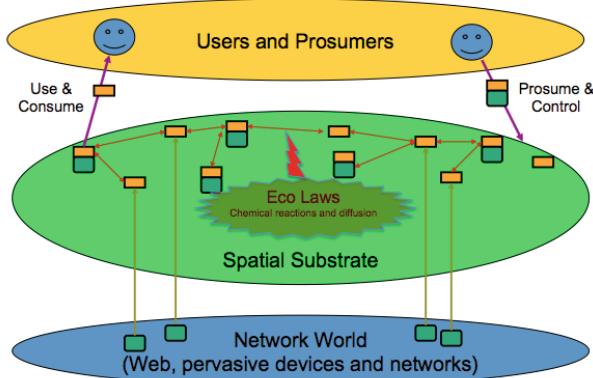


Fig. 1 The SAPERE Reference Architecture

3 The SAPERE Middleware and Its Programming Interface

In this section we overview how SAPERE applications can be programmed, by introducing the API of the SAPERE middleware and exemplifying its usage. Without having the ambition of fully detailing the SAPERE programming approach, we intend to give readers a clue and enable them to better understand the overall SAPERE development methodology.

3.1 The Middleware

The execution of SAPERE applications is supported by a middleware infrastructure [11] which reifies the SAPERE architecture in terms of a lightweight software support, enabling a SAPERE node to be installed in tablets and smartphones. Operationally, all SAPERE nodes (whether fixed at the infrastructure level or mobile) are considered at the same level since the middleware code they run could support the same services and provides the same set of functions.

Each SAPERE node hosts a local tuple space [2], that acts as a local repository of LSAs for local agents, and a local eco-laws engine. The LSA-space of each node is in network with a limited set of neighbor nodes based on spatial proximity relations. Such relations consequently determine the spatial shape of the SAPERE substrate. From the viewpoint of individual agents (that will constitute the basic execution unit) the middleware provides an API to access the local LSA space, to advertise themselves (via the injection of an LSA), and to support the agents' need of continuously updating their LSAs. In addition, such API enables agents to detect local events (as the modifications of some LSAs) or the enactment of some eco-laws on available LSAs.

Eco-laws are realized as a set of rules embedded in SAPERE node. For each node, the same set of eco-laws applies to rule the dynamics between local LSAs (in the form of bonding, aggregation, and decay) and those between non-locally-situated LSAs (via the spreading eco-law that can propagate LSAs from a node to another to support distributed interactions). From the viewpoint of the underlying network infrastructure, the middleware transparently absorbs dynamic changes at the arrival/dismissing of the supporting devices, without affecting the perception of the spatial environment by individuals.

3.2 The SAPERE API

In the SAPERE model, each agent executing on a node takes care of initializing at least one LSA (representing the agent itself), of injecting it on the local LSA space, and of keeping the values of such LSA (and of any additional LSA it decides to inject) updated to reflect its current situation. Each agent can modify only its own LSAs, and eventually read the LSAs to which it has been linked by a proper eco-law. Moreover LSAs can be manipulated by eco-laws, as explained in the following sections.

At the middleware level, a simple API is provided to let agents inject LSA – `injectLSA(LSA myLSA)` – and to let agents atomically update some fields of an LSA to keep it “alive” – `updateLSA(field = new-value)`. In addition, it is possible for an agent to sense and handle whatever events occur on the LSAs of an agent, e.g., some match that triggers some eco-laws. E.g., it is possible to handle the event represented by the LSA being bound with another LSA via the `onBond(LSA mylsa)` method.

The eco-laws assure self-adaptive and self-organizing activities in the ecosystems. Eco-laws operate on a pattern-matching schema: they are triggered by the presence of LSAs matching with each other, and manipulate such LSAs (and the fields within) according to a sort of artificial chemistry [12].

3.3 LSAs

LSAs are realized as descriptive tuples made by a number of fields in the form of “name-value” properties, and possibly organized in a hierarchical fashion: the value of a property can be a property again (called SubDescriptions in SAPERE terms). A detailed description of semantic representation of LSAs is in [9]. Here we emphasize that, by building over tuple-based models and extending upon them [2], the values in a LSA can be: *actual*, yet possibly dynamic and changing over time (which makes LSAs live); *formal* not tied to any actual value unless bound to one and representing a dangling connection (typically represented with a “?”).

Pattern matching between LSAs – which is at the basis of the triggering of eco-laws – happens when all the properties of a description match, i.e., when for each property whose names correspond (i.e., are semantically equivalent) then the associated values match. As in classical tuple-based approaches, a formal value matches with any corresponding actual value.

For instance, the following LSAa: (sensor-type = temperature; accuracy = 0.1; temp = 45), that can express the LSA of a temperature sensor, can match the following LSAb: (sensor-type = temperature; temp = ?), which can express a request for acquiring the current temperature value. LSAa and LSAb match with each other. The properties present in LSAa (e.g., accuracy) are not taken into account by the matching function because it considers only inclusive match.

4 The Eco-laws Set

Let us now detail the SAPERE eco-laws and discuss their role in the SAPERE ecosystem.

4.1 Bonding

Bonding is the primary form of interaction among co-located agents in SAPERE (i.e., within the same LSA space). In particular, bonding can be used to locally discover and access information, as well to get in touch and access local services. All of which with a single and unique adaptive mechanism. Basically, the Bonding eco-law realizes a sort of a virtual link between LSAs, whenever two LSAs (or some SubDescriptions within) match.

The bonding eco-law is triggered by the presence of formal values in at least one of the LSAs involved. Upon a successful pattern matching between the formal values of an LSA and actual values of another LSA, the eco-law creates the bond

between the two. The link established by bonding in the presence of “?” formal fields is bi-directional and symmetric. Once a bond is established the agents holding the LSAs are notified of the new bond and can trigger actions accordingly. After bond creation, the two agents holding the LSAs can read each other LSAs. This implies that once a formal value of an LSA matches with an actual value in an LSA it is bound to, the corresponding agent can access the actual values associated with the formal ones. For instance, with reference to the LSAA and LSAb of the previous subsection, the agent having injected LSAb, upon bonding with LSAA (which the agent can detect with the `onBond` method) it can access the temperature measure by the sensor represented by LSAb.

As bonding is automatically triggered upon match, debonding takes place automatically whenever some changes in the actual “live” values of some LSAs make the matching conditions no longer holding.

In addition to the ? formal field, which establishes a one-to-one bidirectional bond between component, SAPERE also makes it possible to express a “*” formal field, which leads to a one-to-many bond with multiple matchings LSAs. Moreover, the ! formal field expresses a field that is formal unless the other ? field has been bound. This makes it possible for an LSA to express a parameterized services, where the ? formal field represents the parameter of the service, and the ! field represents the answer that it is able to provide once it has been filled with the parameters.

We emphasize that the bonding eco-law mechanism can be used to enable two agents to spontaneously get in touch with each other and exchange information, all of which with a single operation and with both having injected an LSA in the space. And, in the case of the ! field, automatically invoking a service. That is, unlike in traditional discovery of data and services [3], bonding makes possible to compose services without distinguishing between the roles of the involved agents and subsuming the traditionally separated phases of discovery and invocation.

4.2 Aggregate Eco-law

The ability of aggregating information to produce high-level digests of some contextual or situational facts is a fundamental requirement for adaptive and dynamic systems. In fact, in open and dynamic environments, one cannot know *a priori* which actual information will be available (some information source may disappear, other may appear, etc.) and the availability of ways to extract a summary of all available information (without having to explicitly discover and access the individual information sources) is very important.

The aggregation eco-law is intended to aggregate LSAs together so as to compute summaries of the current system’s context. An agent can inject an LSA with the *aggregate* and *type* properties. The aggregate property identifies a function to base the aggregation upon. The type property identifies which LSAs to aggregate. In particular it identifies a numerical property of LSAs to be aggregated. For example `LSAc:(aggregation_op = max; property = temp)` will trigger the aggregation eco-law that selects all the LSAs having a `temp` numerical property,

computes the maximum value among them, and modifies the LSAs with the result. In the current implementation, the aggregation eco-law is capable of performing most common order and duplicate insensitive (ODI) aggregation functions [7].

The aggregation eco-law supports separation of concern and allows to re-use previous aggregations. On the one hand, an agent can request an aggregation process without dealing with the actual code to perform the aggregation. On the other hand, the LSA resulting from an aggregation can be read (via a proper bond) by any other agent that needs to get the pre-computed result.

4.3 Decay Eco-law

The Decay eco-law enables the vanishing of components from the SAPERE environment. The Decay eco-law applies to all LSAs that specify a decay property to update the remaining time to live according to the specific decay function, or actually removing LSAs that, based on their decay property, are expired. For instance in LSAd: (sensor-type = temperature; temp = 10; DECAY=1000) it makes that LSA to be automatically deleted after a second.

The Decay eco-law therefore is a kind of garbage collector capable of removing LSAs that are no longer needed in the ecosystem or no longer maintained by a component, for instance because they are the result of a propagation.

4.4 Spread Eco-law

The above presented eco-laws basically act on a local basis, i.e., on a single LSA space. Since the SAPERE model is based on a set of networked interaction spaces, it is of course fundamental to enable non-local interactions, and specifically by provide a mechanism to send information to remote LSA spaces and make it possible to distribute information and results across a network of LSA spaces.

To this end, in SAPERE we designed a so called “spread” eco-law, capable of diffusing LSAs to remote spaces. One of the primary usages of the spread eco-law is to enable searches for components that are not available locally, and vice versa to enable the remote advertisement of services. For an LSA to be subjected to the spread eco-law, it has to include a `diffusion` field, whose value (along with additional parameters) defines the specific type of propagation.

Two different types of propagation are implemented in the SAPERE framework: (*i*) a direct propagation used to spread an LSA to a specified neighbor node, e.g., LSAe:(...`diffusion_op=direct`; `destination=node_x`; ...); (*ii*) a general diffusion capable of propagating an LSA to all neighboring SAPERE nodes, e.g., LSAf:(...`diffusion_op=general`; `hop = 10`; ...), where the `hop` value can be specified to limit the distance of propagation of the LSA from the source node.

General diffusion of an LSA via the spread eco-law to distances greater than one is a sort of broadcast that induces a large number of replicas of the same LSA to

reach the same nodes multiple times from different paths. To prevent this, general diffusion is typically coupled with the aggregation eco-law, so as to merge together such multiple replicas.

5 From Eco-laws to Distributed Self-organization

The four above presented eco-laws form a necessary and complete set to support self-organizing nature-inspired interactions.

The four eco-laws are necessary to support decentralized adaptive behaviors for pervasive service systems. Bonding is the necessary mean to support adaptive local service interactions, subsuming the necessary phases of discovery and invocation of traditional service systems. Spreading is necessary in that there must be a mean to diffuse information in a distributed environment to enable distributed interactions. Aggregation and decay are necessary to support decentralized adaptive access to information without being forced to dynamically deploy code on the nodes of the system, which may not be possible in decentralized environments.

Further, and possibly of more software engineering relevance, the eco-law set is sufficient to express a wide variety of distributed interaction schemes (or “patterns”), there included self-organizing ones. Bonding and spreading can be trivially used to realize local and distributed client-server scheme of interactions as well as asynchronous models of interactions and information propagation. Coupling spreading with aggregation and decay, however, it is possible to realize also those distributed data structures necessary to support all patterns of nature-inspired adaptive and self-organizing behaviors, i.e., virtual physical fields, digital pheromones, and virtual chemical gradients [1].

In particular, aggregation applied to the multiple copies of diffused LSAs can reduce the number of redundant LSAs so as to form a distributed *gradient* structures, also known as *computational force fields*. As detailed in [5], many different classes of self-organized motion coordination schemes, self-assembly, and distributed navigation can be expressed in terms of gradients. For instance, Figures 2 shows how it is possible to define a “Guide” agent that builds, with its LSA, a distributed computational field and another agent “Search” that follows such field uphill.

In addition, spreading and aggregation can be used together to produce distributed self-organized aggregations, i.e., dynamically computing some distributed property of the system and have the results of such computation available at each and every node of the system, as from [7]. Distributed aggregation is a basic mechanism via which to realize forms of distributed consensus and distributed task allocation and behavior differentiation. For instance, the code in Figure 3 shows how it is possible to aggregate temperature information from multiple distributed sensors.

By bringing also the decay eco-law into play, and combining it with spreading and aggregation, one can realize pheromone-based data structures, which makes possible to realize a variety of bio-inspired schemes for distributed self-organization [1].

```

Agent Guide {
    init() { injectLSA(name = guide, diffusion_op = general,
                        hop = 1, aggregation_op = min, previous = local) } }

Agent Search {
    init() { injectLSA(name = guide, hop = *) }

    onBond(LSA b) { float d = computeDistanceFromHop(b.hop)
                     print("guide distance = "+d)
                     print("go toward "+b.previous) } }

```

Fig. 2 Generating and navigating distributed data structures. The agent **Guide** uses the spread eco-law combined with aggregation to create field-like data structures, that agent **Search** can then detect and follow downhill.

```

Agent X {
    init() { injectLSA(aggregation_op = max, property = temp, diffusion = general,
                      hop = 1, previous = local) }

Sensor 1....N {
    init() { float t = sample()
              injectLSA(temp = t) }

    run() { while(true) { float t = sample()
                           updateLSA(temp = t) }}}}

```

Fig. 3 Distributed aggregation. Many temperature sensors 1--N exist in the ecosystem. An agent **X** can inject an LSA that, by combining spreading and aggregation, can adaptively compute the maximum temperature of sensors.

In particular, while general diffusion and progressive decay can be used to realize diffusible and evaporating pheromone-like data structures, direct propagation can be used to navigate by following pheromone gradients.

6 Related Works and Conclusions

While exploiting a number of common features with situated pervasive approaches based on tuple spaces [6, 5, 4], SAPERE proposes a different concept and management of space. Indeed local spaces in SAPERE cannot merge – as the ones in *Lime* [6] do. Differently from TOTA [5] the behaviors are embedded in the eco-laws rather than in the tuples. Moreover components in a SAPERE environment do not need to declare a personalized view of the context as in *Ego-spaces* [4], that is naturally provided by injecting their own LSAs.

Some approaches, such as *SwarmLinda* [10], exploit the properties of adaptive self-organizing natural systems to enforce adaptive behaviors transparently to applications. SAPERE has the more ambitious goal of promoting and easing the programming of self-organizing distributed behaviors.

Summarizing, the innovative nature-inspired approach of SAPERE is effective to easily enforce a variety of self-organizing schemes for pervasive computing services. As the activities within the SAPERE European Project will proceed, we will

challenge the SAPERE findings and tools against innovative services in the area of crowd management, by exploiting an ecosystem of pervasive displays as a technical testbed [8].

Acknowledgements. Work supported by the EU FET Unit, under grant No. 256873.

References

1. Babaoglu, O., et al.: Design patterns from biology for distributed computing. *ACM Trans. Auton. Adapt. Syst.* 1(1), 26–66 (2006)
2. Gelernter, D.: Generative communication in linda. *ACM Trans. Program. Lang. Syst.* 7(1), 80–112 (1985)
3. Huhns, M.N., Singh, M.P.: Service-oriented computing: Key concepts and principles. *IEEE Internet Computing* 9(1), 75–81 (2005)
4. Julien, C., Roman, G.-C.: Egospaces: Facilitating rapid development of context-aware mobile applications. *IEEE Trans. Software Eng.*, 281–298 (2006)
5. Mamei, M., Zambonelli, F.: Programming pervasive and mobile computing applications: the tota approach. *ACM Trans. Software Engineering and Methodology* 18(4) (2009)
6. Murphy, A.L., Picco, G.P., Roman, G.-C.: Lime: A coordination model and middleware supporting mobility of hosts and agents. *ACM Trans. Software Engineering and Methodology* 15(3), 279–328 (2006)
7. Nath, S., Gibbons, P.B., Seshan, S., Anderson, Z.R.: Synopsis diffusion for robust aggregation in sensor networks. In: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, Baltimore, MD, USA, pp. 250–262 (2004)
8. Sippl, A., Holzmann, C., Zachhuber, D., Ferscha, A.: Real-time gaze tracking for public displays. In: de Ruyter, B., Wichert, R., Keyson, D.V., Markopoulos, P., Streitz, N., Divitini, M., Georgantas, N., Mana Gomez, A. (eds.) *AmI 2010. LNCS*, vol. 6439, pp. 167–176. Springer, Heidelberg (2010)
9. Stevenson, G., Viroli, M., Ye, J., Montagna, S., Dobson, S.: Self-organising semantic resource discovery for pervasive systems. In: 1st International Workshop on Adaptive Service Ecosystems: Natural and Socially Inspired Solutions, Lyon, France, pp. 47–52 (2012)
10. Tolksdorf, R., Menezes, R.: Using swarm intelligence in linda systems. In: Omicini, A., Petta, P., Pitt, J. (eds.) *ESAW 2003. LNCS (LNAI)*, vol. 3071, pp. 49–65. Springer, Heidelberg (2004)
11. Zambonelli, F., Castelli, G., Mamei, M., Rosi, A.: Integrating pervasive middleware with social networks in sapere. In: International Conference on Selected Topics in Mobile and Wireless Networking, Shanghai, PRC, pp. 145–150 (2011)
12. Zambonelli, F., Viroli, M.: A survey on nature-inspired metaphors for pervasive service ecosystems. *Journal of Pervasive Computing and Communications* 7, 186–204 (2011)

SOMEWHERE2 – A Robust Package for Collaborative Decentralized Consequence-Finding

Philippe Chatalic and Andre de Amorim Fonseca

Abstract. This paper presents SOMEWHERE2, a new framework that may be used for setting up peer-to-peer inference systems and for solving consequence finding problems in a completely decentralized way. It is a complete redesign and reengineering of an earlier platform. The new architecture has gained in genericity, modularity and robustness. It is much easier to extend and/or to reuse as a building block for advanced distributed applications, such as Peer Data Management Systems.

1 Introduction

The *consequence finding* problem [9, 10] amounts to finding formulas that are consequences of a logical theory. Many applications involve reasoning tasks that aim at discovering such consequences, not explicit in the original theory. Often, not all consequences are sought, but only a subset of those, satisfying some syntactical property, called a *production field* [12]. Consequence finding is more complex than the *proof finding* problem, for which a user simply wants to *verify* whether a formula is entailed or not by a theory. It has proved to be useful for wide range of problems involving diagnosis, abductive reasoning, hypothetical and non-monotonic reasoning, query rewriting as well as knowledge compilation (see [10] for a survey).

There are several reasons to consider this problem in a distributed setting. For large theories, the problem may rapidly become out of scope for a single computing unit. Exploiting structural properties of the original theory in order to decompose it into subparts is one possible approach. It has been explored in the context of theorem proving by [2] and recently extended to the case of consequence finding in [3].

Philippe Chatalic
L.R.I. - Bat 650, Université Paris-Sud, Orsay, France
e-mail: chatalic@lri.fr

Andre de Amorim Fonseca
L.R.I. , Inria Saclay Ile-de-France, Orsay, France
e-mail: andre.amorimfonseca@gmail.com

But the need for a distributed approach becomes essential when the knowledge is intrinsically scattered at different places. This is the case in some multi agent architectures, where each agent is not necessarily willing (e.g. for some privacy reasons) to share all of its knowledge, but has to collaborate with others in order to achieve its goals. Similarly, semantic data management systems exploit the content of multiple sources of information, each of them being described using its own ontology. Query answering over such networked data sources requires reasoning over distributed ontologies. It generally proceeds in two steps, the first of which is a query rewriting step (that can be reformulated as a consequence finding problem), where the original query is rewritten in terms of the languages of the different relevant ontologies. Obtained rewritings are then evaluated on the appropriate sources.

Given the ever growing number of information sources available over the web, peer-to-peer (P2P) architectures look particularly promising for that purpose. The absence of any centralized control or hierarchical organization and the fact that each peer plays the same role gives much flexibility to accommodate to the dynamic nature of such networks. This also contributes to the scalability and the robustness of such approaches. Such principles are at the core of Peer Data Management Systems (PDMS) such as EDUTELLA [11], PIAZZA [5] or SOMEWHERE [1].

SOMEWHERE is a framework based on a decentralized propositional P2P inference system (P2PIS) that can be used as a corner stone for designing elaborated PDMS. Its scalability on fairly large networks of peers has been very encouraging. It is however rather a *proof of concept* than a rock solid piece of code. Unstable, it missed essential features, e.g. the ability to cope with the dynamicity of the network. Moreover, costs for its maintenance and attempts to add new features turned out to be extremely high. At some time, the best solution has appeared to start a complete reengineering, in order to improve both its design, robustness and extensibility. The main contribution of this paper is to present the core architecture of this new system.

2 Consequence Finding in P2P Inference Systems

SOMEWHERE is based on a decentralized consequence finder that consider P2PIS $\mathcal{P} = \{P_i\}_{i=1..n}$ such as the one of Fig. 1, where each peer has its own *vocabulary* V_i (a set of propositional variables) and a local clausal theory $P_i = O_i \cup M_i$. O_i denotes the set of *local* clauses, that are made exclusively of literals over V_i (here symbols indexed by i), while M_i denotes *mapping* clauses, involving the vocabulary of at least two different peers. Intuitively, local clauses describe the very own knowledge of the peer P_i while mappings state logical constraints between different peer theories. Variables appearing in several peers are said to be *shared* (edges labels on fig. 1). They characterize possible interactions between peers and implicitly define an *acquaintance graph*. We assume each peer to be aware of its acquaintances and denote by $ACQ(l, P_i)$ the set of peers with which P_i shares the variable of a literal l . The *global theory* $P = \bigcup_{i=1..n} P_i$ is a set of clauses over the vocabulary $V = \bigcup_{i=1..n} V_i$.

For such networks, we consider the classical semantics of propositional logic. We use \models to denote the classical consequence relation. A clause c is an *implicate* of

a theory Σ iff $\Sigma \models c$. An implicate c is *prime* iff for any other implicate c'' of Σ , $c'' \models c$ implies $c'' \equiv c'$. By extension a clause c' is said to be a (*prime*) *implicate* of a clause c wrt Σ iff it is a (*prime*) implicate of $\Sigma \cup \{c\}$. Furthermore, c' is said to be a *proper* (*prime*) implicate of c wrt Σ if $\Sigma \cup \{c\} \models c'$, but $P \not\models c'$.

Decentralized Consequence Finding

Given a P2PIS \mathcal{P} , the problem we address is to compute all the *proper prime implicants* of a clause c with respect the global theory P . The point is that while c is stated using the language \mathcal{L}_{V_i} (clauses over V_i) of the queried peer, the proper prime implicants of c can be clauses of \mathcal{L}_V ¹. Moreover, none of the peer in the P2PIS has a global view of the network. A peer only knows its own theory P_i and the variables shared with its neighbours.

DECA [1] is the first sound and complete decentralized algorithm that has been proposed to solve this problem. It proceeds using a split/recombination strategy. When a peer P_i is asked to compute the proper prime implicants of a literal q , it first computes the proper prime implicants of q w.r.t. the local theory P_i . Each implicate c is then split in two subclauses $L(c)$ and $S(c)$, corresponding to the non-shared and shared literals of c . If non empty, $S(c)$ is then split in turn and for each shared literal l of $S(c)$ DECA asks its relevant neighbours $ACQ(l, P_i)$ (which are running the very same algorithm) to compute similarly the proper consequences of l wrt P . Answers of respective calls on neighbours are then recombined incrementally with $L(c)$.

Illustrative Example. The reader is referred to [1] for the full details on the algorithm. But to get an intuition of the work performed by DECA, let us illustrate the reasoning triggered by the query $q_0 = d_4 @ P_4$ that asks P_4 to compute the proper prime implicants of d_4 wrt P , on the P2PIS of Fig. 1. First, local consequents of d_4 on P_4 are computed, which gives : $\{d_4, \neg a_4, \neg b_4 \vee c_4, \neg d_1 \vee c_4\}$. But c_4 and d_1 being shared variables, this triggers two recursive queries, $q_1 = c_4 @ P_3$ and $q_2 = \neg d_1 @ P_1$.

q₁: local consequents of c_4 on P_3 are $\{c_4, \neg d_3\}$. Since d_3 is not shared, the reasoning halts. Both clauses are returned to P_4 , as answers for the query q_1 .

q₂: local consequents of $\neg d_1$ on P_1 are $\{\neg d_1, a_1, b_1 \vee e_1\}$. But e_1 being shared with P_3 , this triggers a new query $q_3 = e_1 @ P_3$.

q₃: local consequents of e_1 on P_3 are $\{e_1, \neg b_3\}$. But b_3 being shared with P_2 , this triggers a new query $q_4 = b_2 @ P_2$.

q₄: local consequents of $\neg b_3$ on P_2 are $\{\square\}$ (i.e. the empty clause, which subsumes all other consequents). \square is then returned to P_3 as an answer for q_4 , where it

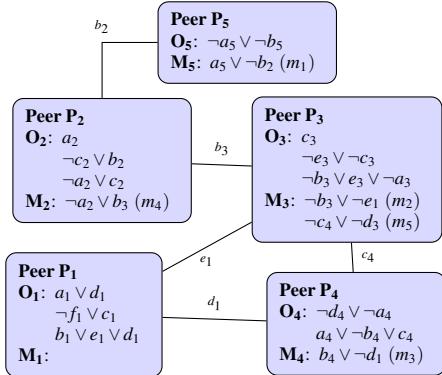


Fig. 1 A P2PIS network

¹ A variant problem is to focus on the proper prime implicants that belong to some production field $PF \subseteq \mathcal{L}_V$, supposed to characterize interesting clauses for the application domain.

subsumes e_1 and $\neg b_3$. \square is then returned to P_1 , as the only answer for the query q_3 .

On P_1 : \square (which subsumes e_1) is recombined with b_1 and the set of consequents of $\neg d_1$ is now $\{\neg d_1, a_1, b_1\}$. This set is returned to P_4 as the answer for the query q_2 .

On P_4 : these clauses are recombined in place of d_1 with the answers obtained for c_4 , producing : $\{\neg d_1 \vee c_4, a_1 \vee c_4, b_1 \vee c_4, \neg d_1 \vee \neg d_3, a_1 \vee \neg d_3, b_1 \vee \neg d_3\}$. These answers are added to those previously obtained, namely $\{d_4, \neg a_4, \neg b_4 \vee c_4, \neg b_4 \vee \neg d_3\}$.

While [1] assume the global theory to be consistent, [4] has adapted this approach to the case where the local theories P_i are consistent, but not necessarily the whole theory P . Two algorithms are described : P2P-NG and WF-DECA, that can respectively detect all causes of inconsistencies and ensure that only so-called *well founded* consequents are produced. Algorithms DECA, P2P-NG and WF-DECA have many similarities but also differences. Their respective codes have been developed by different persons, at different periods of time, with different coding practices and style. Moreover, the original code from which they evolved suffered from many flaws, with lots of duplicated code and serious *cross cutting concerns* that strongly affected its modularity. Prohibitive maintenance costs and difficulties to add new functionalities have motivated a complete reengineering of the whole.

3 Architecture of SOMEWHERE2

SOMEWHERE2's design has been driven by several goals, among which the obtention of more robust and flexible code, developed according better software engineering practices. Robustness has been improved through a careful analysis of the different parts of the code in order to reduce dependencies as much as possible. This has lead to the design of several *components* corresponding to central concepts.

Flexibility has been improved by structuring the code in terms of an abstract notion of *module*. Each module addresses a specific concern. Some *required modules* are always loaded by the application. Others may be included (or not) at build time, according to the user's needs. A *module manager* is responsible for loading the appropriate modules. Essential functionalities of the various components are modeled in *abstract modules* and implemented in *concrete modules*. This module based approach greatly facilitates alternative concrete implementations of functionalities, the selection different sets of features and/or the creation of new extensions.

As seen in the illustrative example, DECA requires a local consequence finder, some way to interact with other peers, to recombine the results obtained from distant peers, to interact with the user (for asking queries, updating/adding/removing peers,...). P2P-NG and WF-DECA have similar needs, although declined in different ways. The current architecture of SOMEWHERE2 (Fig. 2) has 4 components (*Module Manager*, *User interface*, *Transport* and *Distributed Reasoning*), each of which with several modules. SOMEWHERE2's default configuration also rely on the component (*IASI Libs*), that offers reasoning services, but in centralized setting. As it can be used independently of *Somewhere2* it is seen as an external dependency.

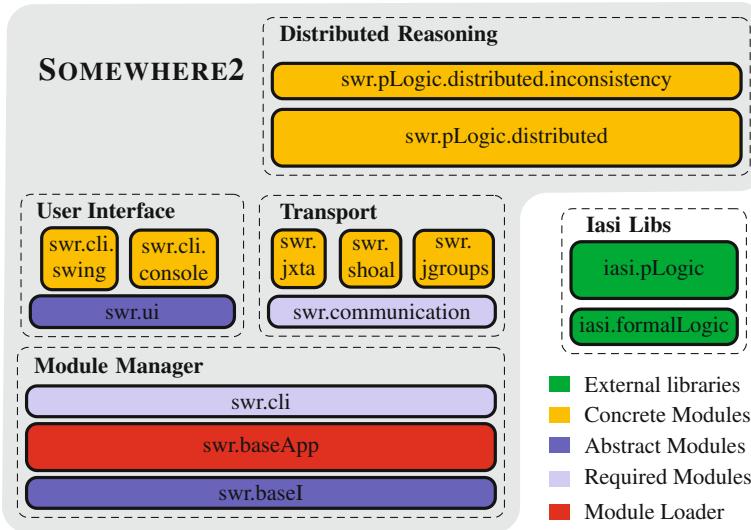


Fig. 2 SOMEWHERE2 Architectural Schema

Components have very few dependencies, represented by top-bottom adjacencies, e.g. *Distributed Reasoning* only depends on *Transport* and *IASI Libs*. Each component can contain required, abstract, concrete and/or external modules. Module dependencies inside a component are reflected in the same way. The *baseApp* module plays a central role and is responsible for the instantiation and configuration of the other modules. Each module can have specific libraries and its own configuration scheme. We briefly describes noticeable features of some of these components.

Transport. In [1], all communications were handled in an ad'hoc way, at socket level. In contrast, the new architecture is designed to reuse an existing P2P framework. The abstract module *communication* describes the concurrency model (based on http-like sessions) and core concepts required by the application for exchanges between peers (messages types, processors, ...) and the dynamicity of the network (joining/leaving peers, lost connections,...). The clean separation of abstract and concrete layers has greatly facilitated the comparison of alternative P2P frameworks (jxta[13], shoal[6] and jgroups[7]), without affecting other parts of the code.

Distributed Reasoning. This component is responsible for all knowledge level concepts relevant to the distributed aspects of consequent finding algorithms [1, 4] (e.g. messages, handlers, network/peers modifications, anytime recombination,...). One module handle all aspects related to inconsistency tolerance and the implementations of P2P-NG and WF-DECA. Both share as much code as possible with DECA, which is implemented in the *pLogic.distributed* module.

IASI Libs. Although packaged as a independent project, this component has been developed simultaneously to the other modules. It is the core library for local CF

algorithms. In contrast, with the [1], that used a simple split/backward chaining strategy, SOMEWHERE2 uses a corrected and optimized version of IPIA [8].

The increased robustness of this new framework also results from a permanent effort to follow good software engineering practices, such as the systematic use of unit tests, the intensive use of design patterns and of static code analyzers (Sonar). A Jenkins server has also been configured to set up integration tests. In its current state, the project represents around 13000 lines of Java code, with less than 5% code redundancy, structured as a set of Maven projects, to ease the build process.

4 Conclusion

We have presented the architecture of SOMEWHERE2. It reunifies in a single and coherent framework two variants, tolerant or not to inconsistent theories of decentralized consequence finding algorithms. This new framework, the code of which has been completely rewritten and reorganized, has gained in modularity, flexibility and robustness. One noticeable improvement is the ability to deal safely with dynamic networks, with peers joining and leaving the network anytime. We expect SOMEWHERE2 to be much easier to use, to maintain and to extend. An extensive experimental study is underway and we plan to release its code under an open source licence.

References

1. Adjiman, P., Chatalic, P., Goasdoué, F., Rousset, M.-C., Simon, L.: Distributed reasoning in a peer-to-peer setting: Application to the semantic web. *JAIR*, 25 (January 2006)
2. Amir, E., McIlraith, S.: Partition-based logical reasoning. In: *KR*, pp. 389–400 (2000)
3. Bourgne, G., Inoue, K.: Partition-based consequence finding. In: *ICTAI*, pp. 641–648 (2011)
4. Chatalic, P., Nguyen, G.H., Rousset, M.C.: Reasoning with Inconsistencies in Propositional Peer-to-Peer Inference Systems. In: *ECAI*, pp. 352–357 (August 2006)
5. Halevy, A.Y., Ives, Z., Tatarinov, I., Mork, P.: Piazza: data management infrastructure for semantic web applications, pp. 556–567. ACM Press (2003)
6. Shoal – a dynamic clustering framework, <http://shoal.java.net>
7. Jgroups - a toolkit for reliable multicast communication,
<http://www.jgroups.org>
8. Kean, A., Tsiknis, G.K.: An incremental method for generating prime impllicants/implicates. *J. Symb. Comput.* 9(2), 185–206 (1990)
9. Lee, C.T.: A completeness theorem and a computer program for finding theorems derivable from given axioms. PhD thesis, Univ. of California, Berkeley, CA (1967)
10. Marquis, P.: Consequence Finding Algorithms. In: *Handbook on Defeasible Reasoning and Uncertainty Management Systems*, vol. 5, pp. 41–145. Kluwer Academic (2000)
11. Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., et al.: Edutella: a p2p networking infrastructure based on rdf, pp. 604–615. ACM (May 2002)
12. Siegel, P.: Représentation et utilisation de la connaissance en calcul propositionnel. PhD thesis, Université d'Aix-Marseille II (1987)
13. Jxta: A language and platform independent protocol for p2p networking,
<http://jxta.kenai.com>

Heuristic Co-allocation Strategies in Distributed Computing with Non-dedicated Resources

Victor Toporkov, Anna Toporkova, Alexey Tselishchev, and Dmitry Yemelyanov

Abstract. In this work, we introduce heuristic slot selection and co-allocation strategies for parallel jobs in distributed computing with non-dedicated and heterogeneous resources (clusters, CPU nodes equipped with multicore processors, networks etc.). A single slot is a time span that can be assigned to a task, which is a part of a job. The job launch requires a co-allocation of a specified number of slots starting synchronously. The challenge is that slots associated with different resources of distributed computational environments may have arbitrary start and finish points that do not match. Some existing algorithms assign a job to the first set of slots matching the resource request without any optimization (the first fit type), while other approaches are based on an exhaustive search. In our approach, co-allocation strategies formalized by given criteria are implemented by algorithms of linear complexity on an available slots number. The novelty of the approach consists of allocating alternative sets of dynamically updated slots based on the information from local resource managers in the node domains. It provides possibilities to optimize job scheduling during resource selection.

Victor Toporkov · Dmitry Yemelyanov
National Research University “MPEI”,
ul. Krasnokazarmennaya, 14, Moscow, 111250, Russia
e-mail: {ToporkovVV, YemelyanovDM}@mpei.ru

Anna Toporkova
National Research University Higher School of Economics,
Moscow State Institute of Electronics and Mathematics,
Bolshoy Trekhsvyatitelsky per., 1-3/12,
Moscow, 109028, Russia
e-mail: atoporkova@hse.ru

Alexey Tselishchev
European Organization for Nuclear Research (CERN),
Geneva, 23, 1211, Switzerland
e-mail: Alexey.Tselishchev@cern.ch

1 Introduction

Economic mechanisms are used to solve problems like resource management and scheduling of jobs in a transparent and efficient way in distributed environments such as cloud computing [8], utility Grid [6], and multi-agent systems [2]. A resource broker model [2, 5, 6, 8] is decentralized, well-scalable and application-specific. It has two parties: node owners and brokers representing users. The simultaneous satisfaction of various application optimization criteria submitted by independent users is not possible due to several reasons and also can deteriorate such quality of service rates as total execution time of a batch of jobs or overall resource utilization. Another model is related to virtual organizations (VO) [7, 14, 15] with central schedulers providing job-flow level scheduling and optimization. VOs naturally restrict the scalability, but uniform rules for allocation and consumption of resources make it possible to improve the efficiency of resource usage and to find a trade-off between contradictory interests of different participants.

In [14, 15], we have proposed a hierarchical scheduling model which is functioning within a VO. The significant difference between the approach proposed in [14, 15] and well-known scheduling solutions for distributed environments such as Grids [3, 5, 6, 7, 16], e.g., gLite Workload Management System [3], is the fact that the *scheduling strategy* is formed on a basis of efficiency criteria. They allow reflecting economic principles of resource allocation by using relevant cost functions and solving a load balancing problem for heterogeneous resources. The metascheduler [14, 15] implements the economic policy of a VO based on local resource schedules. The schedules are defined as sets of slots coming from resource managers or schedulers in the resource domains. During each scheduling cycle the sets of available slots are updated and two problems have to be solved: 1) selecting an alternative set of slots (alternatives) that meet the requirements (resource, time, and cost); 2) choosing a slot combination that would be the efficient or optimal in terms of the whole job batch execution. To implement this scheduling scheme, first of all, one needs an algorithm for finding and co-allocating slot sets. An optimization technique for the second phase of this scheduling scheme was proposed in [14, 15].

First fit selection algorithms [1, 4] assign any job to the first set of slots matching the resource request conditions, while other algorithms use an exhaustive search, feature more than linear complexity, and may be inadequate for on-line use [10]. Moab scheduler [9] implements backfilling and during a slot window search does not take into account any additive constraints such as the minimum required storage volume or the maximum allowed total allocation cost. Moreover, backfilling does not support environments with non-dedicated resources. NWIRE system [5] performs a slot window allocation based on the user defined efficiency criterion under the maximum total execution cost constraint. However, the optimization occurs only on the stage of the best found offer selection.

In our previous works [11, 12, 13], two algorithms for slot selection AMP and ALP that feature linear complexity $O(m)$, where m is the number of available time-slots, were proposed. Both algorithms perform the search of the first fitting window without any optimization. AMP (Algorithm based on Maximal job Price),

performing slot selection based on the maximum slot window cost, proved the advantage over ALP (Algorithm based on Local Price of slots) when applied to the above mentioned scheduling scheme. However, in order to accommodate an end user's job execution requirements, there is a need for more precise slot selection algorithms to consider various user demands along with the VO resource management policy. In this paper, we propose heuristic strategies for effective slot selection and co-allocation based on users or VO administrators defined criteria. The novelty of the proposed approach consists of allocating a number of alternative sets of slots. It makes it possible to optimize job scheduling under specified criteria.

The paper is organized as follows. Sect. 2 introduces a general scheme for searching alternative slot sets that are effective by given criteria. Then some strategies based on this scheme are proposed and considered. Sect. 3 contains simulation results for comparison of proposed and known strategies. Sect. 4 summarizes the paper and describes further research topics.

2 General Scheme and Slot Co-allocation Strategies

In this section we introduce a general scheme of an Algorithm searching for Extreme Performance (AEP) and its implementations as diverse co-allocation strategies formalized by given criteria.

2.1 AEP Scheme

The job is a set of n interrelated tasks. The launch of any job requires a co-allocation of a specified number of slots, as well as in the classic backfilling variation [9]. The target is to scan a list of available slots and to select a window W of n parallel slots with a length of the required resource reservation time. The length of each slot in the window is determined by the performance rate of the resource on which it is allocated. The time length of an allocated window W is defined by the execution time of the task that is using the slowest CPU node, and in the case of heterogeneous resources, as a result one has a window with a “rough right edge”(Fig. 1). One can define a criterion on which the best matching window alternative is chosen. This can be a criterion crW for a minimum cost, a minimum execution runtime or, for example, a minimum energy consumption.

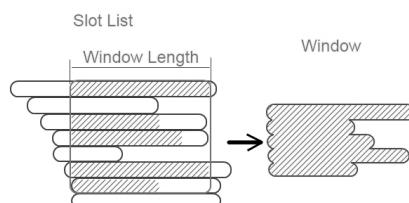


Fig. 1 Window with a “rough right edge”

The scheme for an optimal window search by the specified criterion may be represented as follows:

```

/* Job - Batch job for which the search is performed ;
** windowSlots - a set (list) of slots representing the window; */
slotList = orderSystemSlotsByStartTime();
for(i=0; i< slotList.size; i++){
    nextSlot = slotList[i];
    if(!properHardwareAndSoftware(nextSlot))
        continue; // The slot does not meet the requirements
    windowSlotList.add(nextSlot);
    windowStart = nextSlot.startTime;
    for(j=0; j<windowSlots.size; j++){
        wSlot = windowSlots[j];
        minLength = wSlot.Resource.getTime(Job);
        if((wSlot.EndTime - windowStart) < minLength)
            windowSlots.remove(wSlot);
    }
    if(windowSlots.size >= Job.m){
        curWindow = getBestWindow(windowSlots);
        crW = getCriterion(curWindow);
        if(crW > maxCriterion){
            maxCriterion = crW;
            bestWindow = curWindow;
        }
    }
}
}

```

Finally, a variable `bestWindow` will contain an effective window by the given criterion `crW`.

Consider as an example the problem of selecting a window of size n with a total cost no more than S from the list of $m > n$ slots (in the case, when $m = n$ the selection is trivial). The cost of using each of the slots according to their required time length is: c_1, c_2, \dots, c_m . Each slot has a numeric characteristic z_i in accordance to crW . The total value of these characteristics should be minimized in the resulting window. Then the problem is formulated as follows:

$$\begin{aligned} a_1 z_1 + a_2 z_2 + \dots + a_m z_m &\rightarrow \min, \\ a_1 c_1 + a_2 c_2 + \dots + a_m c_m &\leq S, \quad a_1 + a_2 + \dots + a_m = n, \quad a_r \in \{0, 1\}, r = 1, \dots, m. \end{aligned}$$

Finding the coefficients a_1, a_2, \dots, a_m each of which takes integer values 0 or 1 (and the total number of “1” values is equal to n because each job is the set of n interrelated tasks), determine the window with the specified criterion crW extreme value. Additional restrictions can be added, for example, considering the specified value of deadline.

By combining the optimization criteria, VO administrators and users can form diverse slot selection and co-allocation strategies for every job in the batch. Users may be interested in their jobs total execution cost minimizing or, for example, in the earliest job finish time, and are able to affect the set of alternatives found by specifying the job co-allocation criteria. VO administrators, in their turn, are

interested in finding extreme integral characteristics for job-flows execution (e.g., total cost, total CPU usage time) to form more flexible and, possibly, more effective slot combinations representing batch execution schedules.

2.2 AEP-Based Heuristic Strategies

The need to choose alternative sets of slots for every batch job increases the complexity of the whole scheduling scheme [14]. With a large number of available slots the search algorithm execution time may become inadequate. Though it is possible to mention some typical optimization problems, based on the AEP scheme that can be solved with a relatively decreased complexity. These include problems of total job cost minimizing, total runtime minimizing, the window formation with the minimal start/finish time.

For the proposed AEP efficiency analysis the following heuristic strategies and their algorithmic implementations were added to the simulation model [14, 15].

1. *AMP* – searching for slot windows with the earliest start time. This scheme was introduced in works [11, 12, 13].
2. *MinRunTime* – this strategy performs a search for a single alternative with the minimum execution runtime. Given the nature of determining a window runtime, which is equal to the length of the longest composing slot, the following scheme may be proposed:

```

orderSlotsByCost(windowSlots);
resultWindow = getSubList(0,n, windowSlots);
extendWindow = getSubList(n+1,m, windowSlots);
while(extendWindow.size > 0){
    longSlot = getLongestSlot(resultWindow);
    shortSlot = getCheapestSlot(extendWindow);
    extendWindow.remove(shortSlot);
    if((shortSlot.size < longSlot.size)&&
       (resultWindow.cost + shortSlot.cost < S)) {
        resultWindow.remove(longSlot);
        resultWindow.add(shortSlot);
    }
}

```

As a result, the suitable window of the minimum time length will be formed in a variable `resultWindow`. The algorithm described consists of the consecutive attempts to substitute the longest slot in the current window (the `resultWindow` variable) with another shorter one that will not be too expensive. In case when it is impossible to substitute the slots without violating the constraint on the maximum window allocation cost, the current `resultWindow` configuration is declared to have the minimum runtime.

3. *MinFinish* – searching for alternatives with the earliest finish time. This strategy may be implemented using the runtime minimizing procedure presented above.

Indeed, the expanded window has a start time $tStart$ equal to the start time of the last added suitable slot. The minimum finish time for a window on this set of slots is $(tStart + \text{minRuntime})$, where minRuntime is the minimum window length. The value of minRuntime can be calculated similar to the runtime minimizing procedure described above. Thus, by selecting a window with the earliest completion time at each step of the algorithm, the required window will be allocated in the end of the slot list.

4. *MinCost* – searching for a single alternative with the minimum total allocation cost on the scheduling interval. For this purpose in the AEP search scheme n slots with the minimum sum cost should be chosen. If at each step of the algorithm a window with the minimum sum cost is selected, at the end the window with the best value of the criterion crW will be guaranteed to have overall minimum total allocation cost at the given scheduling interval.
5. *MinProcTime* – this strategy performs a search for a single alternative with the minimum total node execution time defined as a sum of the composing slots' time lengths. It is worth mentioning that this implementation is simplified and does not guarantee an optimal result and only partially matches the AEP scheme, because a random window is selected.
6. *Common Stats, AMP* (further referred to as *CSA*) – the strategy for searching multiple alternatives using *AMP*. Similar to the general searching scheme [11, 12, 13], a set of suitable alternatives, disjointed by the slots, is allocated for each job. To compare the search results with the strategies 1-5, presented above, only alternatives with the extreme value of the given criterion will be selected, so the optimization will take place at the selection process. The criteria include the start time, the finish time, the total execution cost, the minimum runtime and the processor time used.

It is worth mentioning that all proposed AEP implementations have a linear complexity $O(m)$: algorithms “move” through the list of m available slots in the direction of non-decreasing start time without turning back or reviewing previous steps.

3 Experimental Studies of Slot Co-allocation Strategies

The goal of the experiment is to examine AEP implementations: to analyze co-allocation strategies with different efficiency criteria, to compare the results with AMP and to estimate the possibility of using in real systems considering the algorithm execution time for each strategy.

3.1 Simulation Environments

A simulation framework [14, 15] was configured in a special way in order to study and to analyze the strategies presented.

In each experiment a generation of the distributed environment that consists of 100 CPU nodes was performed. The performance rate for each node was generated

as a random integer variable in the interval [2; 10] with a uniform distribution. The resource usage cost was formed proportionally to their performance with an element of normally distributed deviation in order to simulate a free market pricing model [2, 5, 6, 8]. The level of the resource initial load with the local and high priority jobs at the scheduling interval [0; 600] was generated by the hyper-geometric distribution in the range from 10% to 50% for each CPU node. Based on the generated environment the algorithms performed the search for a single initial job that required an allocation of 5 parallel slots for 150 units of time. The maximum total execution cost according to user requirements was set to 1500. This value generally will not allow using the most expensive (and usually the most efficient) CPU nodes. The relatively high number of the generated nodes has been chosen to allow CSA to find more slot alternatives. Therefore more effective alternatives could be selected for the searching results comparison based on the given criteria.

3.2 Experimental Results

The results of the 5000 simulated scheduling cycles are presented in Fig. 2. The obtained values of the total job execution cost are as follows: *AMP* – 1445,2; *minFinish* – 1464,2; *minCost* – 1027,3; *minRuntime* – 1464,9; *minProcTime* – 1342,1; *CSA* – 1352.

Each full AEP-based strategy was able to obtain the best result in accordance with the given criterion: start time (Fig. 2 (a)); runtime (Fig. 2 (b)); finish time (Fig. 2 (c)); CPU usage time (Fig. 2 (d)). Besides, a single run of the AEP-like algorithm had an advantage of 10%-50% over suitable alternatives found by *AMP* with a respect to the specified criterion. According to the experimental results, on one hand, the best scheme with top results in start time, finish time, runtime and CPU usage time was *minFinish*. Though in order to obtain such results the algorithm spent almost all user specified budget (1464 of 1500). On the other hand, the *minCost* strategy was designed precisely to minimize execution expenses and provides 43% advantage over *minFinish* (1027 of 1500), but the drawback is a more than modest results by other criteria considered.

The important factor is a complexity and an actual working time of the algorithms implementing heuristic strategies. Fig. 3 (a) shows the actual algorithms execution time in milliseconds measured depending on the number of CPU nodes. The simulation was performed on a regular PC workstation with Intel Core i3 (2 cores @ 2.93 GHz), 3GB RAM on JRE 1.6, and 1000 separate experiments were simulated for each value of the processor nodes numbers {50, 100, 200, 300, 400}.

The *CSA* strategy has the longest working time that on the average almost reaches 3 seconds when 400 nodes are available. A curve “*CSA per Alt*” in Fig. 3 (a) represents an average working time for the *CSA* algorithm in recalculation for one alternative. AEP-based algorithms feature a quadratic complexity. Fig. 3 (b) presents the algorithms working time in milliseconds measured depending on the scheduling interval length. Overall 1000 single experiments were conducted for each value of the interval length {600, 1200, 1800, 2400, 3000, 3600} and for each considered

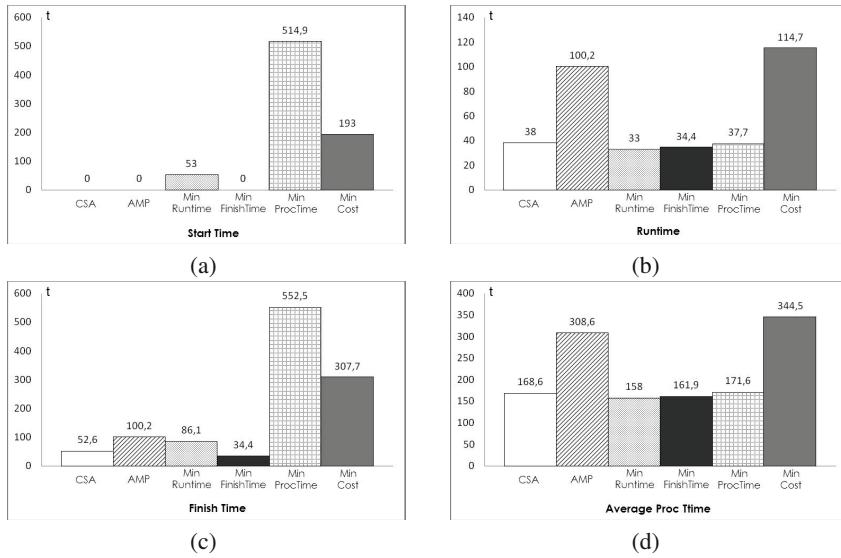


Fig. 2 Average start time (a), runtime (b), finish time (c), and CPU usage time (d)

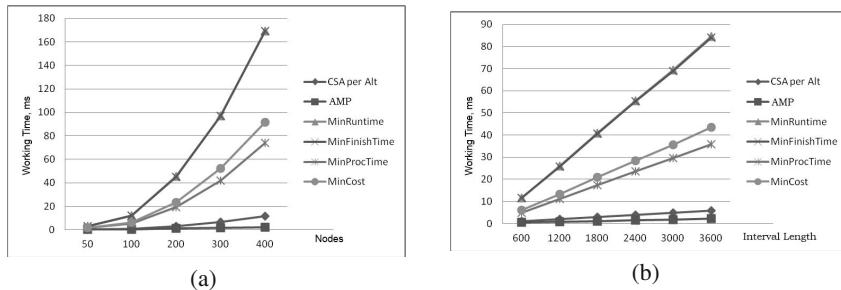


Fig. 3 Average working time duration depending on the available CPU nodes number (a) and the scheduling interval length (b)

algorithm an average working time was obtained. The experiment simulation parameters and assumptions were the same as described earlier in this section, apart from the scheduling interval length. A number of CPU nodes was set to 100. Similarly to the previous experiment, CSA had the longest working time (about 2.5 seconds with the scheduling interval length equal to 3600 model time units), which is mainly caused by the relatively large number of the formed execution alternatives (on the average more than 400 alternatives on the 3600 interval length).

Analyzing the presented values it is easy to see that all proposed algorithms have a linear complexity with the respect to the length of the scheduling interval and, hence, to the number of the available slots.

The *minProcTime* strategy stands apart and represents a class of simplified AEP implementations with a noticeably reduced working time. And though the scheme

compared to other considered algorithms, did not provide any remarkable results, it was on the average only 2% less effective than the CSA scheme by the dedicated CPU usage criterion (see Fig. 2 (d)). At the same time its reduced complexity and actual working time (see Fig. 3(b)) allow to use it in a large wide scale distributed systems when other optimization search algorithms prove to be too slow.

4 Conclusions and Future Work

In this work, we address the problem of slot selection and co-allocation for parallel jobs in distributed computing with non-dedicated resources. For this purpose the AEP scheme as a basis for diverse co-allocation strategies was proposed and considered. Specific AEP heuristic strategies with a reduced complexity were introduced. They include total job cost minimizing, total runtime minimizing, the slot window formation with the minimal start/finish time. Each of the co-allocation algorithms possesses a linear complexity on a total available slots number and a quadratic complexity on a CPU nodes number. The advantage of AEP-based strategies over the general CSA scheme was shown for each of the considered criteria: start time, finish time, runtime, CPU usage time, and total cost. In our further work, we will refine resource co-allocation algorithms in order to integrate them with scalable co-scheduling strategies [14, 15].

Future research will be focused on AEP-based algorithms integration with the whole batch scheduling approach, and mainly on its influence on job-flows execution efficiency.

Acknowledgements. This work was partially supported by the Council on Grants of the President of the Russian Federation for State Support of Leading Scientific Schools (SS-316.2012.9), the Russian Foundation for Basic Research (grant no. 12-07-00042), and by the Federal Target Program “Research and scientific-pedagogical cadres of innovative Russia”(state contract no. 16.740.11.0516).

References

1. Aida, K., Casanova, H.: Scheduling Mixed-parallel Applications with Advance Reservations. In: HPDC 2008, pp. 65–74. IEEE CS Press, New York (2008)
2. Bredin, J., Kotz, D., Rus, D., Maheswaran, R.T., Imer, C., Basar, T.: Computational Markets to Regulate Mobile-Agent Systems. Autonomous Agents and Multi-Agent Systems 6(3), 35–263 (2003)
3. Cecchi, M., Capannini, F., Dorigo, A., et al.: The gLite Workload Management System. J. Phys.: Conf. Ser. 219(6), 062039 (2010)
4. Elmroth, E., Tordsson, J.: A Standards-based Grid Resource Brokering Service Supporting Advance Reservations, Coallocation and Cross-Grid Interoperability. J. of Concurrency and Computation: Practice and Experience 25(18), 2298–2335 (2009)
5. Ernemann, C., Hamscher, V., Yahyapour, R.: Economic Scheduling in Grid Computing. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2002. LNCS, vol. 2537, pp. 128–152. Springer, Heidelberg (2002)

6. Garg, S.K., Buyya, R., Siegel, H.J.: Scheduling Parallel Applications on Utility Grids: Time and Cost Trade-off Management. In: 32nd Australasian Computer Science Conference, pp. 151–159 (2009)
7. Kurowski, K., Nabrzyski, J., Oleksiak, A., Weglarz, J.: Multicriteria Aspects of Grid Resource Management. In: Nabrzyski, J., Schopf, J.M., Weglarz, J. (eds.) Grid Resource Management. State of the Art and Future Trends, pp. 271–293. Kluwer Acad. Publ. (2003)
8. Lee, Y.C., Wang, C., Zomaya, A.Y., Zhou, B.B.: Profit-driven Scheduling for Cloud Services with Data Access Awareness. *J. of Parallel Distributed Computing* 72(4), 591–602 (2012)
9. Moab Adaptive Computing Suite,
<http://www.adaptivecomputing.com/products/>
10. Takefusa, A., Nakada, H., Kudoh, T., Tanaka, Y.: An Advance Reservation-based Co-allocation Algorithm for Distributed Computers and Network Bandwidth on QoS-guaranteed Grids. In: Frachtenberg, E., Schwiegelshohn, U. (eds.) JSSPP 2010. LNCS, vol. 6253, pp. 16–34. Springer, Heidelberg (2010)
11. Toporkov, V., Bobchenkov, A., Toporkova, A., Tselishchev, A., Yemelyanov, D.: Slot Selection and Co-allocation for Economic Scheduling in Distributed Computing. In: Malyshkin, V. (ed.) PaCT 2011. LNCS, vol. 6873, pp. 368–383. Springer, Heidelberg (2011)
12. Toporkov, V., Toporkova, A., Bobchenkov, A., Yemelyanov, D.: Resource Selection Algorithms for Economic Scheduling in Distributed Systems. *Procedia Computer Science* 4, 2267–2276 (2011)
13. Toporkov, V., Yemelyanov, D., Toporkova, A., Bobchenkov, A.: Resource Co-allocation Algorithms for Job Batch Scheduling in Dependable Distributed Computing. In: Zamojski, W., Kacprzyk, J., Mazurkiewicz, J., Sugier, J., Walkowiak, T. (eds.) Dependable Computer Systems. AISC, vol. 97, pp. 243–256. Springer, Heidelberg (2011)
14. Toporkov, V., Tselishchev, A., Yemelyanov, D., Bobchenkov, A.: Composite Scheduling Strategies in Distributed Computing with Non-dedicated Resources. *Procedia Computer Science* 9, 176–185 (2012)
15. Toporkov, V., Tselishchev, A., Yemelyanov, D., Bobchenkov, A.: Dependable Strategies for Job-Flows Dispatching and Scheduling in Virtual Organizations of Distributed Computing Environments. In: Zamojski, W., Mazurkiewicz, J., Sugier, J., Walkowiak, T., Kacprzyk, J. (eds.) Complex Systems and Dependability. AISC, vol. 170, pp. 289–304. Springer, Heidelberg (2012)
16. Yu, J., Buyya, R., Ramamohanarao, K.: Workflow Scheduling Algorithms for Grid Computing. In: Xhafa, F., Abraham, A. (eds.) Metaheuristics for Scheduling in Distributed Computing Environments. SCI, vol. 146, pp. 173–214. Springer, Heidelberg (2008)

Distributed Version of Algorithm for Generalized One-Sided Concept Lattices

Peter Butka, Jozef Pócs, and Jana Pócsová

Abstract. In this paper we provide the distributed version of algorithm for creation of model Generalized One-Sided Concept Lattices (GOSCL), special case for fuzzy version of data analysis approach called Formal Concept Analysis (FCA), which provide the conceptual model of input data based on the theory of one-sided concept lattices and was successfully applied in several domains. GOSCL is able to work with data tables containing the different attribute types processed as fuzzy sets. One problem with the creation of FCA model is computational complexity. In order to reduce the computation times, we have designed the distributed version of the algorithm and showed its applicability on the generated data set. The algorithm is able to work well especially for data where number of newly generated concepts is reduced (like for sparse input data tables).

Peter Butka
Technical University of Košice,
Faculty of Electrical Engineering and Informatics,
Department of Cybernetics and Artificial Intelligence,
Letná 9, 040 01 Košice, Slovakia
e-mail: peter.butka@tuke.sk

Jozef Pócs
Palacký University Olomouc, Faculty of Science,
Department of Algebra and Geometry,
17. listopadu 12, 771 46 Olomouc, Czech Republic, and
Mathematical Institute, Slovak Academy of Sciences,
Grešíkova 6, 040 01 Košice, Slovakia
e-mail: pocs@saske.sk

Jana Pócsová
Technical University of Košice, BERG Faculty,
Institute of Control and Informatization of Production Processes,
Boženy Němcovej 3, 043 84 Košice, Slovakia
e-mail: jana.pocsova@tuke.sk

1 Introduction

The large amount of available data and the needs for their analysis brings up the challenges to the area of data mining. It is evident that methods for different analysis should be more effective and understandable. One of the conceptual data mining methods, called Formal Concept Analysis (FCA, [7]), is an exploratory data analytical approach which identifies conceptual structures (concept lattices) among data sets. FCA has been found useful for analysis of data in many areas like knowledge discovery, data/text mining, information retrieval, etc. The standard approach to FCA provides the method for analysis of object-attribute models based on the binary relation (where object has/not particular attribute). The extension of classic approach is based on some fuzzifications for which object-attribute models describe relationship between objects and attributes as fuzzy relations. From the well-known approaches for fuzzification we could mention an approach of Bělohlávek [2], an approach of Krajčí [12], the approach based on the multi-adjoint concept lattices [13], and also work by one of the authors generalizing the other approaches [15].

In practical applications, so-called one-sided concept lattices are interesting, where usually objects are considered as a crisp subsets (as in classical FCA) and attributes are processed as fuzzy sets. In case of one-sided concept lattices, there is strong connection with clustering (cf. [10]). As it is known, clustering methods produce subsets of a given set of objects, which are closed under intersection, i.e., closure system on the set of objects. Since one-sided concept lattice approach produces also closure system on the set of objects, one can see one-sided concept lattice approaches as a special case of hierarchical clustering. Several one-sided approaches to FCA were already defined, we mention papers of Krajčí [11] and Yahia & Jaoua [1]. These approaches allow only one type of attribute (i.e., truth degrees structure) to be used within the input data table. In our previous paper [3] we have introduced the necessary details and incremental algorithm for model called GOSCL (Generalized One-Sided Concept Lattice), which is able to work with the input data tables with different types of attributes, e.g., binary, quantitative (values from interval of reals), ordinal (scale-based), nominal, etc.

One of the problems of the FCA-based methods is computational complexity, which generally can be (in worst case) exponential. We have analyzed several aspects of GOSCL complexity. In [4] we have shown that for fixed input data table and attributes time complexity of GOSCL is asymptotically linear with the increasing number of objects. This is based on the fact that after some time (which is specific for the data set) new concepts are added linear to the increment of objects. Moreover, in [5] we have analyzed the significant reduction of computation times of the algorithm for sparse input data tables (i.e., input tables with many zeros). However, in order to achieve the objective to produce large-scale concept lattices on real data, which can be then used in retrieval tasks or text-mining analysis, it is possible to extend our approach and re-use distributed computing paradigm based on data distribution [9]. Therefore, the main aim of this paper is to describe the possibility to distribute the computation of GOSCL algorithm for larger context based on its decomposition to several subtables with the separated (and disjoint) subsets of rows,

i.e., data table is decomposed to several smaller tables, for which small lattices are created and these are then iteratively combined (by defined merging procedure) to one large concept lattice for the whole data table. The presented algorithm is also analyzed according to the randomly generated data with different sparseness.

In the following section we provide necessary details for definition of generalization of one-sided concept lattices and the algorithm for their creation. Section 3 is devoted to the introduction of distributed approach for creation of model, also with the detailed description of the algorithm. In next section, some basic experiments are provided in order to show the potential of the algorithm for its future usage on the real data.

2 Generalized One-Sided Concept Lattices

In this section we provide necessary details about the fuzzy generalization of classical concept lattices, so called generalized one-sided concept lattices, which was introduced in [3].

The crucial role in the mathematical theory of fuzzy concept lattices play special pairs of mappings between complete lattices, commonly known as Galois connections. Hence, we provide necessary details regarding Galois connections and related topics.

Let (P, \leq) and (Q, \leq) be complete lattices and let $\varphi: P \rightarrow Q$ and $\psi: Q \rightarrow P$ be maps between these lattices. Such a pair (φ, ψ) of mappings is called a *Galois connection* if the following condition is fulfilled:

$$p \leq \psi(q) \quad \text{if and only if} \quad \varphi(p) \geq q.$$

Galois connections between complete lattices are closely related to the notion of closure operator and closure system. Let L be a complete lattice. By a *closure operator* in L we understand a mapping $c: L \rightarrow L$ satisfying:

- (a) $x \leq c(x)$ for all $x \in L$,
- (b) $c(x_1) \leq c(x_2)$ for $x_1 \leq x_2$,
- (c) $c(c(x)) = c(x)$ for all $x \in L$ (i.e., c is idempotent).

Next we describe mathematical framework for one-sided concept lattices. We start with the definition of generalized formal context.

A 4-tuple (B, A, L, R) is said to be a *one-sided formal context* (or *generalized one-sided formal context*) if the following conditions are fulfilled:

- (1) B is a non-empty set of objects and A is a non-empty set of attributes.
- (2) $L: A \rightarrow \text{CL}$ is a mapping from the set of attributes to the class of all complete lattices. Hence, for any attribute a , $L(a)$ denotes the complete lattice, which represents structure of truth values for attribute a .
- (3) R is generalized incidence relation, i.e., $R(b, a) \in L(a)$ for all $b \in B$ and $a \in A$. Thus, $R(b, a)$ represents a degree from the structure $L(a)$ in which the element $b \in B$ has the attribute a .

Then the main aim is to introduce a Galois connection between classical subsets of the set of all objects $\mathbf{P}(B)$ and the direct products of complete lattices $\prod_{a \in A} \mathsf{L}(a)$ which represents a generalization of fuzzy subsets of the attribute universe A . Let us remark that usually fuzzy subset are considered as function from the given universe U into real unit interval $[0, 1]$ or more generally as a mappings from U into some complete lattice L . In our case the generalization of fuzzy subsets is straightforward, i.e., to the each element of the universe (in our case the attribute set A) there is assigned the different structure of truth values represented by complete lattice $\mathsf{L}(a)$.

Now we provide a basic results about one-sided concept lattices.

Let (B, A, L, R) be a generalized one-sided formal context. Then we define a pair of mapping ${}^\perp : \mathbf{P}(B) \rightarrow \prod_{a \in A} \mathsf{L}(a)$ and ${}^\top : \prod_{a \in A} \mathsf{L}(a) \rightarrow \mathbf{P}(B)$ as follows:

$$X^\perp(a) = \bigwedge_{b \in X} R(b, a), \quad (1)$$

$$g^\top = \{b \in B : \forall a \in A, g(a) \leq R(b, a)\}. \quad (2)$$

Let (B, A, L, R) be a generalized one-sided formal context. Then a pair $({}^\perp, {}^\top)$ forms a Galois connection between $\mathbf{P}(B)$ and $\prod_{a \in A} \mathsf{L}(a)$.

Now we are able to define one-sided concept lattice. For formal context (B, A, L, R) denote $C(B, A, \mathsf{L}, R)$ the set of all pairs (X, g) , where $X \subseteq B$, $g \in \prod_{a \in A} \mathsf{L}(a)$, satisfying

$$X^\perp = g \quad \text{and} \quad g^\top = X.$$

Set X is usually referred as *extent* and g as *intent* of the concept (X, g) .

Further we define partial order on $C(B, A, \mathsf{L}, R)$ as follows:

$$(X_1, g_1) \leq (X_2, g_2) \quad \text{iff} \quad X_1 \subseteq X_2 \text{ iff } g_1 \geq g_2.$$

Let (B, A, L, R) be a generalized one-sided formal context. Then $C(B, A, \mathsf{L}, R)$ with the partial order defined above forms a complete lattice, where

$$\bigwedge_{i \in I} (X_i, g_i) = \left(\bigcap_{i \in I} X_i, ((\bigvee_{i \in I} g_i)^\top)^\perp \right)$$

and

$$\bigvee_{i \in I} (X_i, g_i) = \left(((\bigcup_{i \in I} X_i)^\perp)^\top, \bigwedge_{i \in I} g_i \right)$$

for each family $(X_i, g_i)_{i \in I}$ of elements from $C(B, A, \mathsf{L}, R)$.

At the end of this section we briefly describe an incremental algorithm for creating one-sided concept lattices. By the incremental algorithm we mean the algorithm which builds the model from the input data incrementally row by row, where in every moment current model from already processed inputs is correct concept lattice for such subtable (and addition of new increment means to provide another row and update the last model, i.e., concept lattice). Let (B, A, L, R) be a generalized one-sided formal context. We will use the following notation. For $b \in B$ we denote by

$R(b)$ an element of $\prod_{a \in A} \mathcal{L}(a)$ such that $R(b)(a) = R(b, a)$, i.e., $R(b)$ represents b -th row in data table R . Further, let 1_L denote the greatest element of $L = \prod_{a \in A} \mathcal{L}(a)$, i.e., $1_L(a) = 1_{\mathcal{L}(a)}$ for all $a \in A$.

Algorithm 1. Incremental algorithm for GOSCL

Require: generalized context (B, A, \mathcal{L}, R)

Ensure: set of all concepts $C(B, A, \mathcal{L}, R)$

```

1:  $L \leftarrow \prod_{a \in A} \mathcal{L}(a)$                                 ▷ Direct product of attribute lattices
2:  $I \leftarrow \{1_L\}$                                          ▷  $I \subseteq L$  will denote set of intents
3:  $C(B, A, \mathcal{L}, R) \leftarrow \emptyset$ 
4: for all  $b \in B$  do
5:    $I^* \leftarrow I$                                          ▷  $I^*$  represents “old” set of intents
6:   for all  $g \in I^*$  do
7:      $I \leftarrow I \cup \{R(b) \wedge g\}$                          ▷ Generation of new intent
8:   end for
9: end for
10: for all  $g \in I$  do
11:    $C(B, A, \mathcal{L}, R) \leftarrow C(B, A, \mathcal{L}, R) \cup \{(g^\top, g)\}$ 
12: end for
13: return  $C(B, A, \mathcal{L}, R)$                                 ▷ Output of the algorithm

```

3 Distributed Algorithm for Generalized One-Sided Concept Lattices

In this section we provide the theoretical details regarding the computation of GOSCL model in distributed manner. It means that division to the partitions is defined. The main theorem is proved, which shows that concept lattices created for the partitions are equivalent to the concept lattice created for the whole input formal context. Then the algorithm for our approach to distribution is provided. It is based on the division of data table into binary-like tree of starting subsets (with their smaller concept lattices), which are then combined in pairs (using specified merge procedure) until the final concept lattice is available.

Let $\pi = \{B_i\}_{i \in I}$ be a partition of object set, i.e., $B = \bigcup_{i \in I} B_i$ and $B_i \cap B_j = \emptyset$ for all $i \neq j$. This partition indicates the division of objects in considered object-attribute model, where R_i defines several sub-tables containing the objects from B_i and which yields several formal contexts $C_i = (B_i, A, \mathcal{L}, R_i)$ for each $i \in I$. Consequently we obtain the system of particular Galois connections (\perp_i, \top_i) , between $\mathbf{P}(B_i)$ and $\prod_{a \in A} \mathcal{L}(a)$. Next we describe how to create Galois connection between $\mathbf{P}(B)$ and $\prod_{a \in A} \mathcal{L}(a)$. We use the method of creating Galois connections applicable for general fuzzy contexts, described in [15]. Let $X \subseteq B$ be any subset and $g \in \prod_{a \in A} \mathcal{L}(a)$ be any tuple of fuzzy attributes. We define

$$\uparrow(X)(a) = \bigwedge_{i \in I} (X \cap B_i)^{\perp_i} \quad \text{and} \quad \downarrow(g) = \bigcup_{i \in I} g^{\top_i} \quad (3)$$

Theorem 1. Let $\pi = \{B_i\}_{i \in I}$ be a partition of object set. Then the pair of mappings (\uparrow, \downarrow) defined by (3) and the pair (\perp^\top, \top^\perp) defined by (1) represent the same Galois connection between $\mathbf{P}(B)$ and $\prod_{a \in A} \mathbf{L}(a)$.

Proof. Let $X \subseteq B$ be any subset of object set and $a \in A$ be an arbitrary attribute. Using (3) we obtain

$$\uparrow(X)(a) = \bigwedge_{i \in I} (X \cap B_i)^{\perp_i} = \bigwedge_{i \in I} \left(\bigwedge_{b \in X \cap B_i} R_i(b, a) \right) = \bigwedge_{i \in I} \left(\bigwedge_{b \in X \cap B_i} R(b, a) \right).$$

Since π is the partition of the object set B , we have $X = \bigcup_{i \in I} (X \cap B_i)$. Involving the fact, that meet operation in lattices is associative and commutative, we obtain

$$X^\perp(a) = \bigwedge_{b \in X} R(b, a) = \bigwedge_{b \in \bigcup_{i \in I} (X \cap B_i)} R(b, a) = \bigwedge_{i \in I} \left(\bigwedge_{b \in X \cap B_i} R(b, a) \right),$$

which yields $\uparrow(X) = X^\perp$ for each $X \subseteq B$.

Similarly, we have

$$\downarrow(g) = \bigcup_{i \in I} g^{\top_i} = \bigcup_{i \in I} \{b \in B_i : \forall a \in A, g(a) \leq R_i(b, a)\}.$$

Easy computation shows that this expression is equal to

$$\{b \in B : \forall a \in A, g(a) \leq R(b, a)\} = g^\top,$$

which gives $\downarrow(g) = g^\top$ for all elements $g \in \prod_{a \in A} \mathbf{L}(a)$.

Algorithm 2. Distributed algorithm for GOSCL

Require: generalized context (B, A, \mathbf{L}, R) , $\pi = \{B_i\}_{i=1}^{2^n}$ - partition of B with 2^n parts

Ensure: merged concept lattice $C(B, A, \mathbf{L}, R)$

- 1: **for** $i = 1$ to 2^n **do**
 - 2: $C_i^{(n)} \leftarrow \text{GOSCL}(B_i, A, \mathbf{L}, R_i)$ \triangleright Application of Algorithm 1 on all subcontexts
 - 3: **end for**
 - 4: **for** $k = n$ down to 1 **do**
 - 5: **for** $i = 1$ to 2^{k-1} **do**
 - 6: $C_i^{(k-1)} \leftarrow \text{MERGE}(C_{2i-1}^{(k)}, C_{2i}^{(k)})$ \triangleright Merging of adjacent concept lattices
 - 7: **end for**
 - 8: **end for**
 - 9: **return** $C(B, A, \mathbf{L}, R) \leftarrow C_1^{(0)}$ \triangleright Output of the algorithm
-

The presented theorem is the base for our distributed algorithm. The main aim is effectively create closure system in $\prod_{a \in A} \mathbf{L}(a)$ which represents the set of all intents. Of course, there are more options how to process the distribution itself, i.e.,

Algorithm 3. Procedure MERGE for merging two concept lattices

Require: two concept lattices C_1 and C_2

Ensure: $\text{MERGE}(C_1, C_2)$

```

1:  $I_1, I_2 \leftarrow \emptyset$                                 ▷  $I_1, I_2$  will denote list of intents for  $C_1, C_2$ 
2: for all  $(X, g) \in C_1$  do
3:    $I_1 \leftarrow I_1 \cup \{g\}$                             ▷ Extraction of all intents from  $C_1$ 
4: end for
5: for all  $(X, g) \in C_2$  do
6:   if  $g \notin I_1$  then                                ▷ Check for duplications of intents in  $I_1$ 
7:      $I_2 \leftarrow I_2 \cup \{g\}$                             ▷ Extraction of all intents from  $C_2$ 
8:   end if
9: end for
10:  $I \leftarrow \emptyset$                                      ▷ Set of all newly created intents
11: for all  $g \in I_2$  do
12:   for all  $f \in I_1$  do
13:      $I \leftarrow I \cup \{f \wedge g\}$                       ▷ Creation of new intent
14:   end for
15: end for
16:  $\text{MERGE}(C_1, C_2) \leftarrow C_1$ 
17: for all  $g \in I$  do
18:    $\text{MERGE}(C_1, C_2) \leftarrow \text{MERGE}(C_1, C_2) \cup \{(g^\top, g)\}$     ▷ Addition of new concepts
19: end for
20: return  $\text{MERGE}(C_1, C_2)$                          ▷ Output of the merging procedure

```

how to merger sub-models during the process. We have designed the version of distribution which divides the starting set of the objects into 2^n parts, for which local models are created (in parallel) and then pairs of them are combined (in parallel) into 2^{n-1} , and then this process continues, until we have only one concept lattice at the end. This output should be isomorphic to concept lattice created by the sequential run, because we have shown that previous theorem is valid and we only make partitions accordingly to its assumptions and proof. It is easy to see that pair-based merging leads to the binary tree of sub-lattices, with n representing the number of merging levels. Now, we can provide the algorithm description. The Algorithm 2 is distributed version (with the idea described here) and merge procedure is provided separately in Algorithm 3.

4 Illustrative Experiments with the Generated Data Sets

For our illustrative experiments with the provided algorithm we have produced randomly generated input formal contexts with specified sparseness index of the whole input data table. The reason is simply the natural characteristic of FCA-based algorithms in general (with its exponential complexity in worst cases), i.e., whenever the number of concepts (intents) still grows fast with the number of inputs, this data table has still too much different combinations of values of attributes in it (this is problem especially for fuzzy attributes, which we have here) and merge procedure

is not effective in order to get better times in comparison with incremental algorithm. But in the cases with the set of intents for which number of concepts (intents) is strongly reduced, the provided approach to distribution will lead to the reduction of computation times (merge procedures will not need so much time to create their new lattices). Fortunately, as we are very interested in the application of GOSCL in text-mining domain, and sparse inputs generate much reduced concept lattices (more zeros will produce more similar combinations of values), our distributed algorithm seems to be efficient mainly for very sparse input matrices.

In order to simplify the data preparation (generation) process, we have used one type of attribute in the table (scale-based, with 5 possible values $\{0, 1, 2, 3, 4\}$, where 0 is lowest value). The values are generated from this scale. The generation of sparse data table is based on the sparse factor $s \in [0, 1]$), which indicates the ratio of “zeros” (as a real number between 0 and 1) in data table, i.e., s indicates the level of sparseness of generated data table. For higher s , the number of zeros in input is also higher. Simple mechanism for generation was used with random number generator for decision on adding the zero or some non-zero value according to the selected sparseness. The generated data are then very close to the selected sparseness (especially for larger inputs). We can add that in text-mining domains the sparseness factor is very high (usually more than 0.99). The number of generated objects will be (for all our examples) 8192 (just to simplify its division to partitions which can be merged easily in particular levels, 2^{13} is 8192).

First illustrative experiment shows that the efficiency is different for different sparseness. Therefore, we have tried random contexts for 5 and 7 attributes and different sparseness (from 0.5 to 0.9). The results are shown in the left part of the Figure 1. As we can see, with the higher the sparseness of the data, the ratio of computation times between distributed version and sequential (reduction ratio T_{dis}/T_{seq}) is lower (which means better reduction). Another experiment is based on the analysis of fixed sparseness (0.9), where number of attributes is changing from 5 to 20 and also N (the number of levels in merging) has three different settings (5, 8 and 10). The result is shown on the right part of the Figure 1, where we can see that number of merging levels (and therefore also number of starting partitions 2^n) has not big influence on the reduction ratio. On the other hand, with the increasing number of attributes reduction ratio should increase due to higher number of intents produced by the combination of values for more attributes.

One of our main interests is to analyze data from text-mining domains, which are usually represented by very sparse input tables (e.g., sparseness 0.998 for Reuters dataset with basic preprocessing settings). Therefore, we also add some experiments with the very sparse inputs. First, we have analyzed the reduction ratio for fixed sparseness 0.998 and number of merging levels 5, where number of attributes are changing (from 100 to 500). As we can see on the left side of the Figure 2, reduction ratio increased with the number of attributes, but is still quite significant even for 500 attributes. Moreover, this illustrative experiments were not realized with specialized sparse-based implementation, which will probably keep the ratio under 1 for more attributes. For reference see [6], where we have provided and analyzed specialized sparse-based version of GOSCL with significant reduction of computation times.

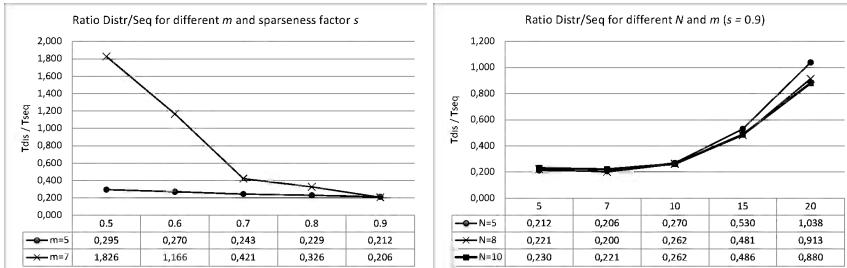


Fig. 1 Experiments with the distributed version of GOSCL - left side: analysis for different sparseness; right side: analysis for different number of attributes and merging levels

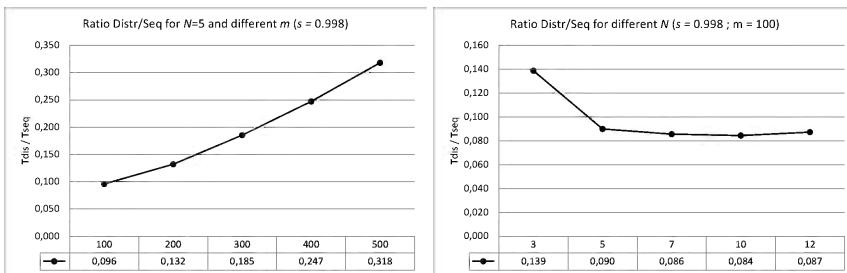


Fig. 2 Experiments with the distributed version of GOSCL on very sparse inputs - left side: analysis of reduction ration with changing number of attributes; right side: analysis of reduction ratio for different number of merging levels

If merge procedure will be implemented (in the future) in similar way, we will be probably able to work with even more attributes with still good reduction ratio. In last illustrative experiment, we have used same sparseness of data (0.998) with 100 attributes and analyze the reduction ratio for different merging levels (N). The result is shown on the right side of the Figure 2, where we can see now more evidently that higher number of levels can help in better times, but it seems that it is not needed to make very small partitions (e.g., for $N = 12$, which leads to starting with two-object sets, ratio is little higher than for previous values). The best number of merging levels can be estimated on real datasets and then used for better results.

At the end of this section we should say that this merging distributed approach seems to be applicable in sparse domains and can lead to computation times reduction, but there are several limitations which should be investigated more on the real data sets. It is expected that also for real data, where randomness is not the way how the table values are produced, the provided algorithm can be useful (but it will depend on data), especially for cases which relatively do not produce too much intents (e.g., very sparse inputs). The first experiments with the Reuters dataset proved the same behavior of the algorithm. Moreover, the usage of specialized sparse-based

implementations should be analyzed in our next experiments. This will show the limits of our distributed GOSCL in creation of large-scale concept lattices from real textual datasets or other sparse-based domains [14].

5 Conclusions

In the presented paper we have introduced the possible solution for distribution of creating the FCA-based model known as Generalized One-Sided Concept Lattice. As we have shown, it is possible to produce merged concept lattice from the smaller ones produced individually for disjoint subsets of objects. For the merging we have introduced simple merging procedure which is based on the partition similar to binary tree (lists are smallest concept lattices from the start of the distribution and root is merged final lattice). The illustrative experiments on the randomly generated data were added which shows its potential (especially for very sparse inputs) and limits. The future work should be done in realization of more experiments (on very sparse real data) and specialized sparse-based implementation of the merging procedure (very first experiments with real dataset proved the results of the paper).

Acknowledgements. This work was supported by the Slovak Research and Development Agency under contracts APVV-0035-10, APVV-0208-10 and APVV-0482-11; by the Slovak VEGA Grants 1/1147/12, 1/0729/12, 1/0497/11; by the ESF Fund CZ.1.07/2.3.00/30.0041.

References

1. Ben Yahia, S., Jaoua, A.: Discovering knowledge from fuzzy concept lattice. In: Kandel, A., Last, M., Bunke, H. (eds.) *Data Mining and Computational Intelligence*, pp. 167–190. Physica-Verlag (2001)
2. Bělohlávek, R.: Lattices of Fixed Points of Fuzzy Galois Connections. *Math. Log. Quart.* 47(1), 111–116 (2001)
3. Butka, P., Pócsová, J., Pócs, J.: Design and Implementation of Incremental Algorithm for Creation of Generalized One-Sided Concept Lattices. In: 12th IEEE International Symposium on Computational Intelligence and Informatics, Budapest, Hungary, pp. 373–378 (2011)
4. Butka, P., Pócsová, J., Pócs, J.: On Some Complexity Aspects of Generalized One-Sided Concept Lattices Algorithm. In: 10th IEEE Jubilee International Symposium on Applied Machine Intelligence and Informatics, Herlany, Slovakia, pp. 231–236 (2012)
5. Butka, P., Pócsová, J., Pócs, J.: Experimental Study on Time Complexity of GOSCL Algorithm for Sparse Data Tables. In: 7th IEEE International Symposium on Applied Computational Intelligence and Informatics, Timisoara, Romania, pp. 101–106 (2012)
6. Butka, P., Pócsová, J., Pócs, J.: Comparison of Standard and Sparse-based Implementation of GOSCL Algorithm. In: 13th IEEE International Symposium on Computational Intelligence and Informatics, Budapest, Hungary, pp. 67–71 (2012)
7. Ganter, B., Wille, R.: *Formal concept analysis: Mathematical foundations*. Springer, Berlin (1999)
8. Grätzer, G.: *Lattice Theory: Foundation*. Springer, Basel (2011)

9. Janciak, I., Sarnovsky, M., Tjoa, A.M., Brezany, P.: Distributed classification of textual documents on the grid. In: Gerndt, M., Kranzlmüller, D. (eds.) HPCC 2006. LNCS, vol. 4208, pp. 710–718. Springer, Heidelberg (2006)
10. Janowitz, M.F.: Ordinal and relational clustering. World Scientific Publishing Company, Hackensack (2010)
11. Krajčí, S.: Cluster based efficient generation of fuzzy concepts. *Neural Netw. World* 13(5), 521–530 (2003)
12. Krajčí, S.: A generalized concept lattice. *Logic Journal of IGPL* 13(5), 543–550 (2005)
13. Medina, J., Ojeda-Aciego, M., Ruiz-Calviño, J.: Formal concept analysis via multi-adjoint concept lattices. *Fuzzy Set. Syst.* 160, 130–144 (2009)
14. Paralić, J., Richter, C., Babić, F., Wagner, J., Raček, M.: Mirroring of Knowledge Practices based on User-defined Patterns. *J. Univers. Comput. Sci.* 17(10), 1474–1491 (2011)
15. Pócs, J.: Note on generating fuzzy concept lattices via Galois connections. *Inform. Sci.* 185(1), 128–136 (2012)

Scalable Spatio-temporal Analysis on Distributed Camera Networks

Kirak Hong, Beate Ottenwälter, and Umakishore Ramachandran

Abstract. Technological advances and the low cost of cameras enable the deployment of large-scale camera networks in airports and metropolises. A well-known technique, called spatio-temporal analysis, enables detecting anomalies on the large-scale camera networks by automatically inferring locations of occupants in real-time. However, spatio-temporal analysis requires a huge amount of system resources to analyze a large number of video streams from distributed cameras. In particular, state update becomes a bottleneck because of the computation and communication overhead of updating a large application state. In this paper we propose a system design and mechanisms for scalable spatio-temporal analysis on camera networks. We present a distributed system architecture including smart cameras and distributed worker nodes in the cloud to enable real-time spatio-temporal analysis on large-scale camera networks. Furthermore we develop selective update mechanisms to improve scalability of our system by reducing the communication cost for state update.

1 Introduction

As cameras are becoming more capable and widely deployed, new application scenarios arise, requiring an automated processing of the continuous video streams to identify and track human beings in real-time. Airports, as an example, currently have more than 1,000 cameras in place and plan to keep increasing the number [1]. In such an environment, a common requirement for *situation awareness applications* [6] is to detect security violations, such as an unauthorized personnel entering a restricted area, by automatically analyzing video streams from distributed

Kirak Hong · Umakishore Ramachandran

College of Computing, Georgia Institute of Technology, Atlanta, Georgia, USA
e-mail: {khong9, rama}@cc.gatech.edu

Beate Ottenwälter

Institute of Parallel and Distributed Systems, University of Stuttgart, Stuttgart, Germany
e-mail: beate.ottenwaelder@ipvs.uni-stuttgart.de

cameras in real-time. Situation awareness applications often rely on a technique called *spatio-temporal analysis* to answer queries related to occupants and zones such as “When did person O leave zone Z?”. Recently, Menon et al. [5] showed the feasibility of spatio-temporal analysis using a global *application state* that is a state transition table representing probabilistic locations of occupants over time.

However, a naive implementation of the spatio-temporal analysis causes poor scalability because a centralized server keeping the global application state becomes a significant performance bottleneck. In this work, we propose a scalable approach to support spatio-temporal analysis on large-scale camera networks. Specifically, our contributions are a) the distributed system design for spatio-temporal analysis, b) identifying the performance bottleneck of spatio-temporal analysis on large-scale camera networks, and c) scalable state update mechanisms to improve overall scalability of spatio-temporal analysis.

Section 2 explains details of spatio-temporal analysis on camera networks and identifies potential bottlenecks in large scale scenarios. Section 3 describes the key problem in terms of system scalability. Section 4 provides our solution, namely, scalable state update to overcome the performance bottleneck. Section 5 presents our evaluation results. Section 6 presents related work and Section 7 presents concluding remarks and future work.

2 Spatio-temporal Analysis

In this section, we discuss the common steps in spatio-temporal analysis and propose a distributed system model for large-scale spatio-temporal analysis. Spatio-temporal analysis is an inference technique that generates probabilistic locations of occupants at a given time using sensor streams. Figure 1 shows an example of spatio-temporal analysis on a camera network. In the example, the face of a person (*signature*) is captured by a camera, from which an *event* is generated. The event is a vector of probabilities representing how closely this signature matches to other pre-registered signatures of known identities in the system. The event causes a transition of an application’s global state to represent up-to-date location of individual occupants. Before making a transition, the previous state is recorded with a timestamp to answer time-dependent queries in the future.

The probabilistic locations captured by a global state is a key to answering various queries referring to location-, occupancy-, and time-dependent questions. Each row in the global state, called an *occupant state*, represents probabilistic location of a specific occupant that allows to answer a *location query* such as “where is occupant O1?”. Each column is called a *zone state*, representing the probabilistic occupancy of a zone that allows to answer *occupancy query* such as “who are presently in zone z1?”. A sequence of global states tagged by timestamp allows *temporal query* such as “When did person B leave zone X?”. By combining the location, occupancy, and temporal queries, application can infer various situations:

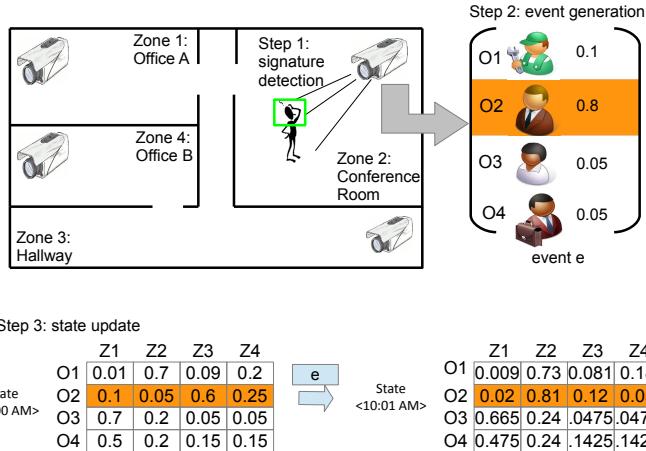


Fig. 1 Example of spatio-temporal analysis on camera networks

“Where did person A and B meet for the last time?”

“How many times did person A move from zone X to zone Y?”

“How many people access zone X today?”

Maintaining the global state involves three steps of processing. The first step, signature detection, involves video analytics to detect signatures such as faces. For example, when a person enters Zone 2 in Figure 1, a face detection algorithm reports the person’s face by analyzing video frames from a camera observing the zone. The second step, event generation, generates a vector of probabilistic estimates about the identity of the detected signature, called an *event*. Depending on the application, different algorithms can be used in this step to generate the vector. For example, various face recognition algorithms [10] or human gait recognition [8] may be used to generate an event, which includes similarities between the detected signature and known signatures. The final step, state update, uses the generated event to update the global state of an application. For example, Figure 1 shows that an occupant *O*2 was in zone *Z*2 with probability 0.05 before the state update, but the probability is increased to 0.8 after making a transition based on an event from *Z*2 with a high probability for *O*2 being in that zone.¹

3 Problem Description

In large-scale spatio-temporal analysis, each processing step involves significant amount of computation and communication, which require distributed computing resources to provide real-time situation awareness. The workload of signature detection and event generation are massively parallel, since each signature and the associated event can be independently computed on distributed nodes including smart cameras and cloud computing resources. This makes the two steps linearly scalable

¹ The concrete formulas to calculate the probability are presented in [5].

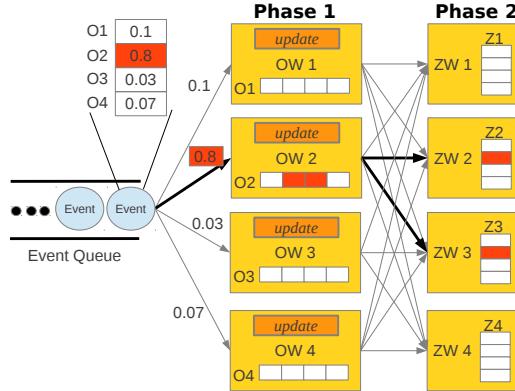


Fig. 2 Distributed state update: For each event, occupant states are updated in parallel at distributed occupant state workers (Phase 1). Once all occupant states are updated, occupant state workers transmit the elements of occupant states to zone state workers (Phase 2). Thin gray lines indicates the communication path for naive state update while thick black lines show communication path for selective update.

with the amount of distributed computing resources. However, unlike the previous two steps, state update requires sequential processing of events due to the inherent nature of maintaining a single global application state. Due to the probabilistic nature of the global application state, an event update potentially affects the probabilities of every occupant in all zones. To allow the temporal evolution of the global application state, each event has to be applied sequentially in temporal order to the current global state. In a large-scale situation awareness application, a centralized approach that maintains the global state at a single node will become a performance bottleneck due to the computation and communication overhead of state update. If many events are generated at the same time, a single node cannot receive and process all events in real-time and therefore the latency for situation awareness will increase. In the following sections, we describe our scalable state update solution to solve this performance bottleneck.

4 Scalable State Update

In order to ensure the scalability of spatio-temporal analysis, both computation and communication overhead of state update must be addressed. To address computation overhead, we propose a distributed state update using partitioned application state across multiple state worker nodes where state update is performed on the partial application states at each node. Figure 2 shows our distributed state update using multiple occupant state workers (OW) and zone state workers (ZW). Each state worker maintains a set of occupant states and zone states to answer queries regarding specific occupants and zones. For example, occupant state worker OW1 maintains the occupant state of O1 to answer location queries on the occupant, while a zone

state worker ZW1 maintains the zone state of Z1 to answer occupancy queries for the zone.

When a new event is generated, each probability for different occupants in the event are delivered to different occupant state workers commensurate with the occupant states that each is responsible for. Upon the arrival of each event element, state update is performed on each occupant state at different occupant state workers (phase 1). Once the occupant states are updated, each occupant state worker transmits elements of occupant states to zone state workers (phase 2). As shown in the figure, the real work of computing the new probabilities for each occupant in every zone is carried out by the occupant state workers in phase 1. Phase 2 is a simple data copy of the computed probabilities in phase 1 and involves no new computation. Since no computation happens at zone state workers, phase 2 may seem to be optional. However, the zone state workers are crucial to handle occupancy queries efficiently because a user has to broadcast a zone-related occupancy query to all occupant state workers if there are no zone state workers.

Although computation overhead is distributed over multiple nodes, communication overhead of state update is still a significant bottleneck. As Figure 2 indicates, each element of an event has to be transmitted to all occupant state workers, which increases the number of messages when more occupant state workers are used. Furthermore, each occupant state worker has to communicate to all zone state workers to update the zone states in phase 2. The total communication cost in terms of bytes transferred linearly depends on the number of occupants and zones in a system:

$$Cost_{comm} = (N_{occupants} + N_{occupants} \times N_{zones}) \times sizeof(double) \quad (1)$$

For example, assuming thousand occupants and thousand zones, and 8 bytes double-precision floating-point representation for event probabilities, each event represents a communication payload of eight megabytes for state update. Assuming hundred signatures are captured from distributed cameras every second, state update would incur a communication cost of eight hundred megabytes per second.

To solve the communication overhead, we propose selective update mechanisms for state update. In a realistic application scenario, the event generation algorithm (e.g., face recognition) may generate an event that has only a few significant probabilities. If an event generation algorithm is highly accurate, i.e., giving high probability for the ground truth identity and very small probability for other identities, a threshold can be applied to use only meaningful event elements for updating specific occupant states. Similarly, another threshold can be applied to occupant states to allow occupant state workers to transmit only significant changes in occupant states to zone state workers. Highlighted elements and arrows in Figure 2 show an example of selective state update. In the example, our system selects only one event element for O2 with significant probability, which is used for state update. After updating an occupant state of O2, only two significant changes for zone Z2 and Z3 are selected and transmitted to corresponding zone state workers. As shown in the figure, the communication cost of state update depends on selected occupants in

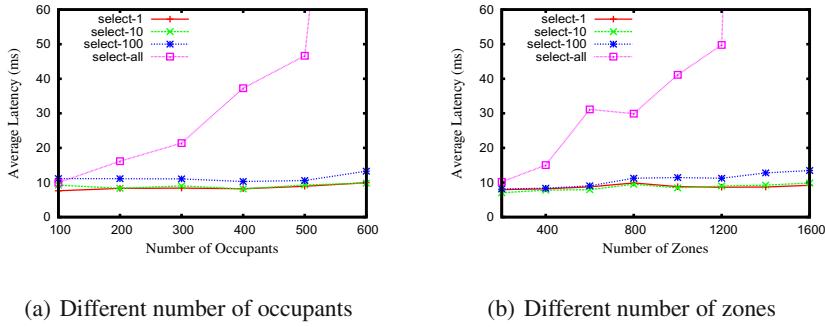


Fig. 3 Average latency for state update with different number of occupants and zones. (a) 1000 zones are used where different number of occupants are selected (b) 425 occupants are used where different number of zones are selected

each event and selected zones from each occupant state rather than the total number of occupants and zones in the system.

To allow such selective state update, our system provides two parameters to users: *occupant selectivity* and *zone selectivity*. Occupant selectivity allows a user to specify the number of occupants to be selected from each event, while zone selectivity specifies the number of zones to be selected from each occupant state. When an event is generated, our system finds occupants with top N probabilities pertaining to the occupant selectivity. Similarly, when an occupant state is updated, our system calculates the difference between the current state and previous state to selects zones with top N changes. Our system supports automatic tuning of an occupant selectivity or a zone selectivity, which makes sure that the total communication cost is bounded by a user-provided threshold. To use the automatic tuning, a user specifies one selectivity (either occupant or zone selectivity) and the maximum communication cost. Using Equation 1, our system automatically infers the right value for an unspecified selectivity to make sure the total communication cost stays below the given maximum communication cost. While event rate changes over time, our system adaptively changes the unspecified selectivity to help system running in real-time in the presence of highly varying event rates from the real world.

5 Evaluation

In this section, we conduct two experiments to show the impact of our approach on scalability and application-level accuracy of spatio-temporal analysis.

5.1 Scalability of State Update

To measure the performance based on a realistic workload, we performed the state update algorithm reported by Menon et al. [5] on eight distributed worker nodes

in Amazon Elastic Compute Cloud (EC2) where each node is an *m1.medium* class. Figure 3(a) shows average latency of state update per event with different selectivity of occupants, while varying the scale of the system in terms of the number of occupants in the system. For example, *select-1* indicates selecting only a single occupant from each event while probabilities for all other occupants are ignored. Similarly, *select-all* indicates that probabilities for all occupants are used for state update. As shown in the figure, the naive state update selecting all event elements (*select-all*) scales poorly, as the system is overloaded when there are more than 500 occupants in the system. Until 500 occupants, latency for state update increases depends on the total number of occupants in the system. Other selective mechanisms show good scalability, where the average latency depends on the number of selected occupants rather than the total number of occupants in the system.

Similarly, Figure 3(b) shows the poor scalability of naive state update and improved scalability and latencies with selective zones for state update. With more than 1200 zones, the naive state update becomes overloaded and cannot handle incoming events in real-time. Other selective state update mechanisms scale well, while the latency for state update depends on the number of zones selected from occupant states. Because we used virtual machines in EC2, available bandwidth and communication latency between distributed state workers vary over time, which results in slightly nonlinear latencies in figures.

5.2 Impact of Selective Update on Spatio-temporal Queries

In this experiment, we show the impact of selective state update on spatio-temporal queries using simulated events that are generated from simulation of moving occupants in camera network². Since selective update ignores small probabilities in events and negligible changes in occupant states, resulting application state can differ from the application state computed by the naive state update. Although the naive state update does not always guarantee correct answers due to its inherently probabilistic nature, we use the naive state update as a baseline to compare with our approximated application state resulting from selective state update.

In this experiment, we used two different types of queries. First type of query, called *location query*, asks whereabouts of a particular occupant. For instance, an application may ask top three most likely places for an occupant in order to select potential video streams to track the occupant. Another type of query, called *occupancy query* asks the occupants who are likely (with higher than 0.5 probability) to be in a specific zone. Using the two types of queries, we compare an approximate application state calculated by selective state update to an application state calculated by the naive state update. For each state update, we issue location queries and occupancy queries for all occupants and zones on the original application state and the approximate application state. If results for the same query differ between the

² We simulated randomly moving occupants in a camera network with a grid topology while events pertain higher probability for a ground truth occupant and lower random probabilities for others.

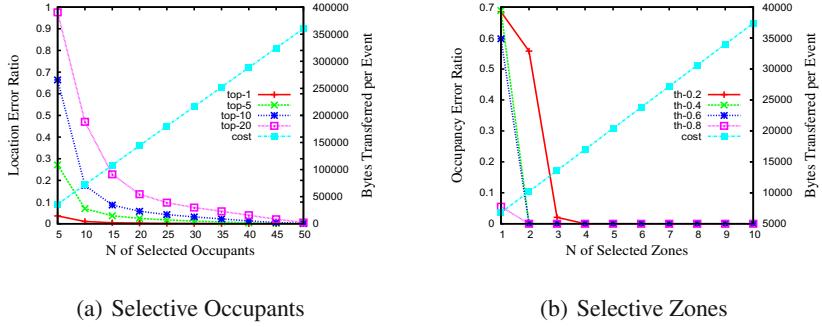


Fig. 4 Accuracy of query results and communication cost when selective state update is used

original and approximate states, we count it as an error. Finally, we calculate the ratio of errors over all the query results.

Figure 4(a) shows the impact of selective update on the result of location queries. This experiment includes four different types of location queries asking different number of probable locations, which can be used for different application scenarios. For instance, *top-1* asks the most likely location for an occupant while *top-10* asks ten probable locations for an occupant to increase the chance to find the occupant. Over all types of location queries, the error ratio reduces when more occupants are selected for state update. However, the communication cost also increases due to the increased number of messages transmitted from an event queue to the occupant state workers. When more probable locations are asked, the error ratio is higher since there is a higher chance of disparity between query results from an approximate state and the original state.

Figure 4(b) shows the error ratio for occupancy queries that ask probable occupants in a specific zone. For occupancy queries, we use a different threshold to answer probable occupancy, ranging from 0.2 to 0.8. Similar to the previous experiment, error ratio reduces when more number of zones are selected for state update while communication cost linearly increases.

Our experimental results shown in Figure 4 indicate that using a small number of occupants and zones for selective state update can significantly reduce the communication overhead without significantly affecting the accuracy as measured by the error rates for the approximate state compared to the original state. For instance, if an application is interested in only the most probable location for each occupant, using only ten occupants for the selective update is sufficient to achieve the same accuracy as compared to the naive state update.

6 Related Work

Many distributed systems help developing large-scale situation awareness applications on camera networks. IBM S3 [2] provides a middleware for video-based smart surveillance system including video analytics and storage modules. Target

Container [3] provides a parallel programming model and runtime system to help domain experts to develop large-scale surveillance application on distributed smart cameras. Above systems focused on processing raw video streams on distributed nodes, which is complementary to our system since we focus on collective inference on events that are generated from video streams.

Moving object databases [9] allow for spatio-temporal analysis by keeping track of mobile devices, like vehicles, and their location. Similar our system, they can handle spatio-temporal queries based on data collected from location sensors [4, 7]. However, unlike events from video streams, the uncertainty of location sensor data (e.g., GPS) are locally confined therefore does not require a global application state.

7 Conclusion

In this paper we addressed the scalability problems of spatio-temporal analysis on large-scale camera networks. Because of the probabilistic nature of event generation and state update, an application has to maintain a global application state to keep track of known occupants in the observed area. We identified state update as the real performance bottleneck of spatio-temporal analysis, and presented solutions to solve the bottleneck. To address the computation overhead of state update, we have presented a distributed state update with a partitioned application state over distributed nodes. To reduce the communication overhead of state update, we have proposed selective state update mechanisms. Our experimental results show that we can effectively remove the performance bottleneck of spatio-temporal analysis by applying selective state update while maintaining similar level of accuracy with the original application states.

References

1. Schiphol Airport leverages technology to drive efficiency,
<http://www.securitysystemsneweurope.com/?p=article&id=se200906VF2ISW>
2. Feris, R., Hampapur, A., Zhai, Y., Bobbitt, R., Brown, L., Vaquero, D., Tian, Y., Liu, H., Sun, M.T.: Case-Study: IBM smart surveillance system. In: Ma, Y., Qian, G. (eds.) Intelligent Video Surveillance: Systems and Technologies. Taylor & Francis, CRC Press (2009)
3. Hong, K., Smaldone, S., Shin, J., Lillethun, D., Iftode, L., Ramachandran, U.: Target container: A target-centric parallel programming abstraction for video-based surveillance. In: 2011 Fifth ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC), pp. 1–8 (2011)
4. Kalashnikov, D.V., Ma, Y., Mehrotra, S., Hariharan, R.: Index for fast retrieval of uncertain spatial point data. In: Proc. of Int'l Symposium on Advances in Geographic Information Systems (ACM GIS 2006), Arlington, VA, USA (2006)
5. Menon, V., Jayaraman, B., Govindaraju, V.: The Three Rs of Cyberphysical Spaces. Computer 44, 73–79 (2011)

6. Ramachandran, U., Hong, K., Iftode, L., Jain, R., Kumar, R., Rothermel, K., Shin, J., Sivakumar, R.: Large-scale situation awareness with camera networks and multimodal sensing. *Proceedings of the IEEE* 100(4), 878–892 (2012)
7. Trajcevski, G., Ding, H., Scheuermann, P., Cruz, I.: Bora: Routing and aggregation for distributed processing of spatio-temporal range queries. In: *2007 International Conference on Mobile Data Management*, pp. 36–43 (2007)
8. Wang, L., Tan, T., Ning, H., Hu, W.: Silhouette analysis-based gait recognition for human identification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25(12), 1505–1518 (2003)
9. Wolfson, O., Xu, B., Chamberlain, S., Jiang, L.: Moving objects databases: issues and solutions. In: *Proceedings of the Tenth International Conference on Scientific and Statistical Database Management*, pp. 111–122 (1998)
10. Zhao, W., Chellappa, R., Phillips, P.J., Rosenfeld, A.: Face recognition: A literature survey. *ACM Comput. Surv.* 35(4), 399–458 (2003)

Service-Wide Adaptations in Distributed Embedded Component-Based Systems

Luís Nogueira and Jorge Coelho

Abstract. Self-adaptive mechanisms are crucial to the design of distributed component-based systems that react to events in a timely and predictable manner to guarantee Quality of Service (QoS) constraints imposed by users, the environment, or applications, despite the uncertain factors that trigger the occurrence of changes in the environment. How these components should interact in order to collectively adapt their behaviour and maintain system-wide properties in a dynamic environment is the focus of this paper.

1 Introduction

The complexity of modern software systems is constantly increasing as new and more complex functionality is added to existing software. An appealing way to reduce such complexity is to apply a component-based design methodology in which the system can be seen as a set of interacting components, each providing a well-defined subset of functionalities and whose integration produces the final system behaviour.

However, the adoption of a component-based approach to embedded software development has been significantly slower than to software development in general. Embedded systems, unlike most general-purpose computing systems, often perform computations subject to various constraints, such as processor speed, amount of memory, power consumption, and reaction time [17]. The timing requirements are often of special importance, particularly for safety-critical systems.

Luís Nogueira
ISEP/CISTER/INESC-TEC, Porto, Portugal
e-mail: lmn@isep.ipp.pt

Jorge Coelho
ISEP/LIACC, Porto, Portugal
e-mail: jmn@isep.ipp.pt

On the other hand, while the general-purpose computing community has begun to embrace the importance and potential of runtime adaptation, the need to adapt to the runtime environment has long been considered especially important in the case of embedded systems. It is widely accepted that the constraints of embedded systems can be significantly reduced by dynamically adapting the system according to the currently required needs and changes in the execution environment.

However, a distributed system composed by self-adaptive components that optimise their behaviour towards some individual goals may not necessarily be optimised at a global level [6]. Complex interdependencies may exist among components such that the incorporation of one change can require the inclusion of several others for the change to work correctly. Complex problems may result from these chain reactions like infinite triggering of new adaptations or inconsistent configurations in different components [16], interference between the different self-management behaviours of components, conflicts over shared resources, sub-optimal system performance and hysteresis effects [6].

Coordination is then a key concept for developing self-adaptive distributed systems [7] and a wide spectrum of coordination strategies have been proposed. However, the limited applicability of existing coordination models to heterogeneous distributed real-time systems [5] provides the significant motivation for the development of a decentralised coordination model that also reasons about the duration and overhead of the coordination process. This paper builds upon the work presented in [12] and proposes an improved and more efficient algorithm to reduce the time and complexity of the needed interactions among components until a collective adaptation behaviour is determined. By exchanging feedback on the desired self-adaptive actions, components converge towards a global solution, even if that means not supplying their individually best solutions. As a result, each component, although autonomous, is influenced by, and can influence, the behaviour of other components in the system.

2 Coordinating Service-Wide Adaptations

Service-oriented applications increasingly consist of a set of interacting software components $S = \{c_1, c_2, \dots, c_n\}$ that jointly provide some service to end-users and operate in open dynamic environments [8]. Components use results produced by other components, that may be running on other nodes, to produce their own output and hence are interdependent. Therefore, as components adapt and evolve, we face the problem of preserving an accurate and consistent model of the service's architecture and its constituent parts. Even though the properties that have to be considered for specifying a global adaptation depend on the particular application and system, there are some general questions that often arise: (i) can a certain configuration be reached?; (ii) are there dependencies between configurations of different components?; (iii) how long will it take to complete an adaptation?; (iv) will it terminate at all?

Taking inspiration from natural self-organising systems is a successful strategy to solve computational problems in distributed systems. In nature there is a well-known, pervasive notion that satisfies these desirable properties of coordination and can be successfully applied to decentralised software systems [3]: the notion of feedback. Nature provides plenty of examples of cooperative self-adaptive and self-organising systems, some of them are far more complex than distributed systems we design and build today [4]. These natural systems are often decentralised in such a way that participants do not have a sense of the global goal, but rather it is the interaction of their local behaviour that yields the global goal as an emergent property. Typically, when feedback is interpreted as negative feedback, it will act as a regulator on changes that deviate the system from some optimal state, whereas feedback perceived as positive will usually trigger adaptation actions that serve to increase changes in the same direction as the previous one.

These feedback loops can provide a generic mechanism for self-adaptation in distributed embedded systems. The adjustment in the output QoS of an individual component $c_i \in S$ is a reflexive action to changes in resource availability. Since the need for coordination arises from conflict-causing interdependencies, the proposed coordination algorithm uses these interdependencies to determine what, when, and to whom to communicate changes in current commitments.

Interdependency relationships among components of a service S can be represented as a directed acyclic graph (DAG) $\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S)$, where each vertex $v_i \in \mathcal{V}_S$ represents a component $c_i \in S$ and a directed edge $e_i \in \mathcal{E}_S$ from c_j to c_k indicates that c_k is functionally dependent on c_j . Within $\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S)$, we give particular names to three types of components. A *source component* models an input device and is not a consumer of the output produced by any other component in the service's DAG. A *sink component* represents a component that is not a producer of any output consumed by other components in \mathcal{G}_S and models a unit to display the global service's output in the end-user's device. Source and sink components mark the limit of a set of components that must be managed in a coordinated manner. Finally, we call *cut-vertex* to a component $c_i \in \mathcal{V}_S$, if the removal of that component divides \mathcal{G}_S in two separate connected graphs. Cut-vertices may confine coordination operations to a subset of \mathcal{G}_S . Within a feasible QoS region, it may be possible to maintain the current output quality by compensating for a decrease in input quality by an increase in the amount of used resources or vice versa [19].

Whenever a component's context changes, it has to decide whether or not it needs to adapt, which may prompt the need for coordination activities that ensure a new globally acceptable solution for the entire distributed service. Therefore, whenever Q_{val}' , the proposed change of the currently output QoS Q_{val}^i for a component $c_i \in S$, has an impact on the currently output QoS level of direct neighbours in \mathcal{G}_S , a request for coordination in the adaptation process is sent to those affected components. Naturally, the formulation of the corresponding positive or negative feedback depends on the feasibility of the new requested QoS level as a function of the quality of the new set of inputs I_{c_j} for component c_j and the amount of locally available resources.

Definition 1. Given a node n and a set of QoS levels σ_n to be provided for all the components being locally executed at n , it is considered that admission control is

performed, and that therefore a system specific *feasibility* function (*e.g.* [9, 1, 14]) determines if a set of QoS requirements can be met with available resources.

$$\text{feasibility}(\sigma_n) = \begin{cases} \text{true} & \text{if node } n \text{ has sufficient resources} \\ & \text{to supply the set of QoS levels } \sigma \\ \text{false} & \text{otherwise} \end{cases}$$

If there are insufficient resources to accommodate the change to the new requested QoS level, a negative feedback is formulated and sent back in reply and the global QoS adaptation fails. On the other hand, positive changes in interdependent commitments are propagated along \mathcal{G}_S , until the next cut-vertex c_c is reached. For that, we define the *paths* and the *flatten* functions. For the sake of simplicity of presentation, we present all the functions in a declarative notation with the same operational model as a pattern matching-based functional language. The reader should note that although a node is only aware of its nearest neighbours in a coalition of cooperating nodes for the collective execution of service S , we deal with the complete interdependency graph only to prove that all the conducted actions are correctly propagated until the final result is determined.

The *paths* function is a breadth first approach with cycle checking for determining components in paths. Visited components are added to the temporary set T in order to avoid infinite loops. The function outputs all the components in all the possible paths between two interdependent components c_i and c_j , or \perp if there is no path between those two components. If there are more than one path between them, the result is a set of sets, each of them corresponding to a distinct path. The *flatten* function is then used to make the resulting set of sets flat, meaning that all the components in these subsets will now belong to a simplified single set.

Definition 2. Given a DAG $\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S)$ and two components $c_i, c_j \in \mathcal{V}_S$, all the components in the possible paths between c_i and c_j are obtained as the result of the function:

$$\begin{aligned} \text{paths}(c_i, c_j, \mathcal{G}_S) &= \text{flatten}(\text{paths}(c_i, c_j, \mathcal{G}_S, \emptyset)) \\ \text{paths}(c_i, c_j, \mathcal{G}_S, T) &= \emptyset, \text{ if } c_i = c_j \\ \text{paths}(c_i, c_j, \mathcal{G}_S, T) &= \{\{c_i, c_{k_1}\} \cup \text{paths}(c_{k_1}, c_j, \mathcal{G}_S, T \cup \{c_{k_1}\}), \\ &\quad \vdots \\ &\quad \{c_i, c_{k_n}\} \cup \text{paths}(c_{k_n}, c_j, \mathcal{G}_S, T \cup \{c_{k_n}\})\}, \\ &\quad \forall c_{k_m} \in \mathcal{V}_S, \text{ such that } (c_i, c_{k_m}) \in \mathcal{E}_S \text{ and } c_{k_m} \notin T \\ \text{paths}(c_i, c_j, \mathcal{G}_S, T) &= \perp, \text{ otherwise} \end{aligned}$$

Definition 3. Given a set A containing other sets, the function *flatten*(A) is defined as:

$$\begin{aligned} \text{flatten}(\emptyset) &= \emptyset \\ \text{flatten}(A) &= \{a\} \cup \text{flatten}(A \setminus \{a\}), \text{ if } a \in A \end{aligned}$$

Affected components collect relevant data, both from the commitments of other components and from local resource reservations, that reflect the current state of

the system. Subsequently, each involved component analyses the collected data and takes a decision on whether and how to adapt in order to reach a global desired state. Finally, to implement the decision, the set of components acts in a coordinated manner. A coordinated adaptive phase is initiated whenever a coordination phase is successful, *i.e.* whenever the component that initiated the coordination request receives a positive feedback in reply. During the coordination phase, each node pre-reserves the needed resources to achieve the new output QoS. Therefore, resource allocation is always possible at this stage [10]. Once resources are allocated, the node commits to produce the announced output QoS either until the service terminates or adaptation occurs.

Definition 4. Given a connected graph $\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_W)$, such that component $c_i \in \mathcal{V}_S$, and $I_{c_i} = \{(c_j, Q_{val}^j), \dots, (c_k, Q_{val}^k)\}$ as the current set of QoS inputs of c_i , and given T as the set of changed QoS inputs in response to the coordination request, the function $update(I, T)$ updates I with the elements from T :

$$\begin{aligned} update(\emptyset, T) &= \emptyset \\ update(I, T) &= \{(c_i, Q_{val'}^i)\} \cup update(I \setminus (c_i, Q_{val}^i), T), \text{ if } (c_i, Q_{val}^i) \in I \\ &\quad \text{and } (c_i, Q_{val'}^i) \in T \\ update(I, T) &= \{(c_i, Q_{val}^i)\} \cup update(I \setminus (c_i, Q_{val}^i), T), \text{ if } (c_i, Q_{val}^i) \in I \\ &\quad \text{and } (c_i, Q_{val'}^i) \notin T \end{aligned}$$

Definition 5. Given the connected graph $\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S)$ with a set of ordered cut-vertices C_S and the subgraph that connects component c_i to next cut-vertex $c_c \in C_S$, the function $test_change(c_i, c_c, \mathcal{G}_S, Q_{val'}^i)$ is defined by:

$$\begin{aligned} test_change(c_i, c_c, \mathcal{G}_S, Q_{val'}^i) &= T, \text{ if} \\ &\quad T' = \{(c_i, Q_{val'}^i)\}, \\ &\quad P = paths(c_i, c_c, \mathcal{G}_S), \\ &\quad I_{QoS} = get_input_qos(c_i), \\ &\quad test_change_component(I_{QoS}, P, T') = T, \\ &\quad T \neq \perp \\ test_change(c_i, c_c, \mathcal{G}_S, Q_{val'}^i) &= \perp, \text{ otherwise.} \end{aligned}$$

$$\begin{aligned} test_change_component(I_{QoS}, \emptyset, T') &= \emptyset \\ test_change_component(I_{QoS}, P, T') &= test_change_component(I_{QoS}, P \setminus \{c_j\}, T') \\ &\quad \cup \{(c_j, Q_{val'}^j)\}, \text{ if} \\ &\quad c_j \in P, \\ &\quad S = update(I_{QoS}, P, T'), \\ &\quad test_feasibility(c_j, Q_{val'}^j, S) = TRUE, \\ test_change_component(I_{QoS}, P, T') &= \perp, \text{ otherwise} \end{aligned}$$

Furthermore, we use the following auxiliary functions which, for the sake of space limitations, we do not define formally:

test_feasibility($c_i, Q_{val'}^i, I_{c_i}$): Given a node n_x where a component $c_i \in S$ is currently being executed, $Q_{val'}^i$ as the new requested QoS level for c_i , and $I_{c_i} = \{(c_j, Q_{val'}^j), \dots, (c_k, Q_{val'}^k)\}$ as the new set of QoS levels given as input to c_i , *test_feasibility* refers to the system's specific feasibility function given by Definition 1 applied to node n_x to determine if the new set of QoS requirements can be met with available resources.

set_qos_level($Q_{val'}^i, c_i, S$): Sets the new QoS level $Q_{val'}^i$ for component $c_i \in S$

get_cut_vertices(\mathcal{G}_S): Returns an ordered set of all the cut-vertices in \mathcal{G}_S . This function is based on the depth-first algorithm for finding cut-vertices which was first presented in [21]. The cut-vertices are found and stored ordered through the DAG from a source component until the sink component.

get_input_qos(c_i): Given a component $c_i \in S$, returns the set of elements (c_j, Q_{val}^j) , where each of these elements represents a component c_j with an output QoS level of Q_{val}^j used as an input in component c_i .

get_qos_level(c_i): Returns the current QoS level output by component c_i .

head(S): Returns the first element of set S .

Having this background, given the connected graph $\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S)$, an end-user sink component c_u , the set of components S and the set of cut-vertices C_S computed by function *get_cut_vertices(\mathcal{G}_S)*, whenever a component $c_i \in \mathcal{V}_S$ is able to change its output QoS to a new QoS level $Q_{val'}^i$, subsequent components in \mathcal{G}_S respond to the coordination request according to a decentralised feedback-based coordination protocol, formally presented in Algorithm 1.

Algorithm 1. Service coordination

$service_coordination(c_i, c_u, \mathcal{G}_S, C_S) = change(F, S)$, if $try_coordination(c_i, C_S \cup \{c_u\}) = F$ and $F \neq \perp$	
$try_coordination(c_i, \emptyset)$ $try_coordination(c_i, C_S)$	$= \emptyset$ $= try_coordination(c_c, C'_S) \cup T$, if $c_c = head(C_S \setminus \{c_i\})$, $Q_{val'}^i = get_qos_level(c_i)$, $test_change(c_i, c_c, \mathcal{G}_S, Q_{val'}^i) = T$, $T \neq \perp$ $= \perp$, otherwise.
$change(\emptyset, S)$ $change(F, S)$	$= TRUE$ $= change(F', S)$, where $(c_j, Q_{val'}^j) \in F$, $set_qos_level(Q_{val'}^j, c_j, S)$, $F' = F \setminus \{(c_j, Q_{val'}^j)\}$.

By exchanging feedback on the performed self-adaptations, components converge towards a global solution, overcoming the lack of a central coordination and global knowledge. Negative feedback loops occur when a change in one service component triggers an opposing response that counteracts that change at other interdependent component along \mathcal{G}_S . On the other hand, positive feedback loops promote global adaptations. The snowballing effect of positive feedback takes an initial change in one component and reinforces that change in the same direction at all the affected partners, requiring only one negotiation round between any pair of interdependent components.

A fundamental advantage of the proposed coordination model is that both the communication and adaptation overheads may depend on the size of a component neighbourhood until a cut-vertex is reached, instead of the entire set of components, if a cut-vertex is able to maintain its current output quality, despite the changes on the quality of its inputs. This allows the proposed feedback-based coordination model to scale effectively to large distributed systems.

3 Properties of the Coordination Model

Proposition 1. *Given a node n and a set of QoS levels σ to be satisfied, the function $\text{feasibility}(\sigma_n)$ always terminates and returns true if σ is feasible in n or false otherwise.*

Proposition 2. *Given a DAG $\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S)$ and two components $c_i, c_j \in \mathcal{V}_S$, $\text{paths}(c_i, c_j)$ terminates and returns all the components in the possible paths between c_i and c_j , \emptyset in case $c_i = c_j$, or \perp in case there is no path between $c_i, c_j \in \mathcal{V}_S$.*

Proposition 3. *Given two sets I and T , both with elements of the form (c_i, Q_{val}^i) , $\text{update}(I, T)$ terminates and returns a new set with the elements of I such that whenever $(c_i, Q_{\text{current}}^i) \in I$ and $(c_i, Q_{\text{new}}^i) \in T$ the pair stored in the returned set is (c_i, Q_{new}^i) .*

Lemma 1. *Given the connected graph $\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S)$ with a set of cut-vertices C_S and the subgraph that connects component c_i to next cut-vertex $c_c \in C_S$ and a new upgraded QoS level value Q_{val}^i , the call to $\text{test_change}(c_i, c_c, \mathcal{G}_S, Q_{\text{val}}^i)$ terminates and succeeds if c_c is able to output a new QoS level Q_{val}^c or fails otherwise.*

Theorem 1 (Correctness of Service Coordination). *Given the connected graph $\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S)$ representing the QoS interdependencies of a service S being executed by a group of components, such that $c_u \in \mathcal{V}$ is the end-user sink component receiving the service at a QoS level Q_{val} , whenever a component c_i announces an upgrade to Q_{val}^i , Algorithm 1 changes the set of SLAs at components in \mathcal{G} such that c_u receives S changed to the QoS level Q_{val}^u or does not change the set of local SLAs at any node and c_u continues to receive S at its current QoS level Q_{val}^u .*

Proof. Termination comes from the finite number of elements in $C_S \cup c_u$ and from Lemma 1. Algorithm 1 applies the function test_change iteratively to all nodes in the subgraph starting with c_i and finishing in c_u . The base case is when there are no

cut-vertices and there is only one call to *test_change*. It is trivial to see that the result of *test_change* will consist in a set of components that will upgrade for the new QoS level Q_{val}^j or it will fail and, by Lemma 1, it is correct. The remaining cases happen when there are one or more cut-vertices between c_i and c_u . Here, *upgrade* will be applied to all subgraphs starting in c_i and finishing in c_u . Each of these subgraphs are sequentially tested. Only if all of them can be upgraded will service S be delivered to component c_u at the new upgraded QoS level Q_{val}^u . The result follows by induction in the number of cut-vertices. \square

Definition 6. Given a directed graph $\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S)$, the in-degree of a component $c_i \in \mathcal{V}$ is the number of edges that have c_i as their destination.

Definition 7. Given a directed graph $\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S)$, the out-degree of a component $c_i \in \mathcal{V}$ is the number of edges that have c_i as their starting node.

Whenever a change to a new QoS level Q_{val}^i is requested by a component c_i , if the next cut-vertex c_c in \mathcal{G} cannot supply the requested level, then all the precedent components between c_i and c_c are kept in their currently supplied feasible QoS level Q_{val}^j . Thus, the number of needed messages is given by the in-degree and out-degree of the nodes in the paths between c_i and c_c where it was determined that the requested new QoS level was not possible. On the other hand, if the requested change is possible, the number of needed messages is given by the number of edges between c_i and the end-user sink component c_u . This is because a change is only possible after all the involved components are queried and the conjunction of their efforts results in a newer QoS level being delivered to c_u . Thus, the maximum number of exchanged messages in a coordination operation is given by Formula 1.

$$\sum_{c \in \mathcal{V}_S} (\deg^+(c) + \deg^-(c)) \quad (1)$$

Applications with real-time requirements can benefit from decentralised coordination mechanisms, as long as those mechanisms support the timing and QoS requirements of applications. Therefore, coordination tasks need to be time dependent since handling time is necessary to specify (and enforce) given levels of quality of service [20, 10].

Access to the system's resources can be modelled as a set of isolated constant-bandwidth servers, either related to CPU [2, 11] or network [13, 15] scheduling. Furthermore, feedback can be formulated as a result of a local anytime QoS adaptation process that can trade off the needed computation time by the achieved solution's quality, within an useful and bounded time [9].

Based on these guarantees, it is possible to determine a time-bounded convergence to a globally accepted solution. As such, the uncertain outcome of iterative decentralised control models whose effect may not be observable until some unknowable time in the future [18] is not present in the proposed regulated coordination model.

Proposition 4. *Given the connected graph $\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S)$ representing the QoS interdependencies of a service S , such that $c_u \in \mathcal{V}$ is the end-user sink component receiving the service at a QoS level Q_{val} , if a source component starts a service*

coordination process involving all the nodes in the graph, this means that for a graph of n components we have to exchange the number of messages given by Formula 1. If the longest exchange of messages between two components takes t_m time units and the longest time for a node to compute its feedback is δ then the convergence of the system for the worst case scenario is given by the formula:

$$\left(\sum_{c \in V_S} (\deg^+(c) + \deg^-(c)) \right) * t_m * \delta \quad (2)$$

4 Conclusions

Dynamic adaptation is frequently used in embedded systems to react to fundamental changes in the environment. However, the adaptation behaviour of distributed embedded systems significantly complicates their design and poses several challenges. In particular, the adaptation of a single component can cause a chain reaction of adaptations in other components that need to be coordinated to maintain desired system-wide properties.

The proposed decentralised coordination approach is based on an effective feedback mechanism to reduce the time and complexity of the needed interactions among components until a collective adaptation behaviour is determined. Feedback is formulated at each affected component as a result of a local QoS adaptation process that evaluates the feasibility of the new requested service solution. This way, positive feedback is used to reinforce the selection of the new desired global service solution, while negative feedback will act as a regulator on changes that deviate the system from some near-optimal state. Therefore, although each individual component has no global knowledge about the group of components cooperatively executing the service and its interdependencies as a whole (only knows to which direct neighbour(s) it sends its output), complex coordinated adaptations emerge from local interactions.

Acknowledgements. This work was partially supported by National Funds through FCT (Portuguese Foundation for Science and Technology) and by the EU ARTEMIS JU funding, within ENCOURAGE project, ref. ARTEMIS/0002/2010, JU grant nr. 269354. Jorge Coelho was partially funded by LIACC through Programa de Financiamento Plurianual of FCT.

References

1. Abdelzaher, T.F., Atkins, E.M., Shin, K.G.: Qos negotiation in real-time systems and its application to automated flight control. *IEEE Transactions on Computers, Best of RTAS 1997 Special Issue* 49(11), 1170–1183 (2000)
2. Abeni, L., Buttazzo, G.: Integrating multimedia applications in hard real-time systems. In: *Proceedings of the 19th IEEE Real-Time Systems Symposium*, Madrid, Spain, p. 4 (1998)
3. Brun, Y., et al.: Engineering self-adaptive systems through feedback loops. In: Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.) *Software Engineering for Self-Adaptive Systems*. LNCS, vol. 5525, pp. 48–70. Springer, Heidelberg (2009)

4. Camazine, S., Deneubourg, J.L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E.: Self-Organization in Biological Systems. Princeton Studies in Complexity. Princeton University Press (2002)
5. Farinelli, A., Rogers, A., Petcu, A., Jennings, N.R.: Decentralised coordination of low-power embedded devices using the max-sum algorithm. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, vol. 2, pp. 639–646 (2008)
6. Friday, A., Davies, N., Cheverst, K.: Utilising the event calculus for policy driven adaptation on mobile systems. In: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks, p. 13. IEEE Computer Society, Washington, DC (2002)
7. Gelernter, D., Carriero, N.: Coordination languages and their significance. Communications of the ACM 35(2), 96–107 (1992)
8. Jin, J., Nahrstedt, K.: Qos-aware service management for component-based distributed applications. ACM Transactions on Internet Technology 8(3), 14:1–14:31 (2008)
9. Nogueira, L., Pinho, L.M.: Dynamic qos adaptation of inter-dependent task sets in cooperative embedded systems. In: Proceedings of the 2nd ACM International Conference on Autonomic Computing and Communication Systems, Turin, Italy, p. 97 (2008)
10. Nogueira, L., Pinho, L.M.: Time-bounded distributed qos-aware service configuration in heterogeneous cooperative environments. Journal of Parallel and Distributed Computing 69(6), 491–507 (2009)
11. Nogueira, L., Pinho, L.M.: A capacity sharing and stealing strategy for open real-time systems. Journal of Systems Architure 56(4-6), 163–179 (2010)
12. Nogueira, L., Pinho, L.M., Coelho, J.: A feedback-based decentralised coordination model for distributed open real-time systems. Journal of Systems and Software 85(9), 2145–2159 (2012)
13. Nolte, T., Lin, K.J.: Distributed real-time system design using cbs-based end-to-end scheduling. In: Proceedings of the 9th International Conference on Parallel and Distributed Systems, pp. 355–360 (2002)
14. Park, J., Ryu, M., Hong, S.: Deterministic and statistical admission control for qos-aware embedded systems. Journal of Embedded Computing 1, 57–71 (2005)
15. Pedreiras, P., Gai, P., Almeida, L., Buttazzo, G.: Ftt-ethernet: a flexible real-time communication protocol that supports dynamic qos management on ethernet-based systems. IEEE Transactions on Industrial Informatics 1(3), 162–172 (2005)
16. Rajkumar, R., Lee, C., Lehoczky, J., Siewiorek, D.: A resource allocation model for qos management. In: Proceedings of the 18th IEEE Real-Time Systems Symposium, p. 298. IEEE Computer Society (1997)
17. Rasche, A., Poize, A.: Dynamic reconfiguration of component-based real-time software. In: Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, pp. 347–354 (2005)
18. Serugendo, G.D.M.: Handbook of Research on Nature Inspired Computing for Economy and Management. In: Autonomous Systems with Emergent Behaviour, pp. 429–443. Idea Group, Inc., Hershey (2006)
19. Shankar, M., de Miguel, M., Liu, J.W.S.: An end-to-end qos management architecture. In: Proceedings of the 5th IEEE Real-Time Technology and Applications Symposium, pp. 176–191. IEEE Computer Society, Washington, DC (1999)
20. Stankovic, J., Abdelzaher, T., Lu, C., Sha, L., Hou, J.: Real-time communication and coordination in embedded sensor networks. Proceedings of the IEEE 91(7), 1002–1022 (2003)
21. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM J. Comput. 1(2), 146–160 (1972)

Distributed/Parallel Genetic Algorithm for Road Traffic Network Division for Distributed Traffic Simulation

Tomas Potuzak

Abstract. In this paper, a distributed/parallel method for division of road traffic networks for distributed road traffic simulation is described. The method is based on its sequential version, which we developed during our previous research. This sequential version utilizes the weights of traffic lanes representing the numbers of vehicles moving within them and a genetic algorithm for the division of the road traffic network into the required number of load-balanced sub-networks interconnected with minimal number of divided traffic lanes. The distributed/parallel version of the division method described in this paper uses a similar approach, but utilizes a distributed/parallel computing environment for a distributed/parallel execution of the genetic algorithm and, consequently, for the speedup of the entire method.

1 Introduction

A utilization of a distributed computing environment where combined power of multiple interconnected computers is utilized simultaneously is a commonly used approach for the speedup of a detailed road traffic simulation [1]. An example of a distributed computer can be a cluster of ordinary interconnected desktop computers (e.g. in a classroom of a university). Today, these computers often incorporate multi-core processors, which enable to perform several processes or threads concurrently and, consequently, to achieve additional speedup of the simulation [2].

For the distributed simulation of a road traffic network, it is necessary to divide this network into required number of sub-networks, which are then simulated on particular nodes of the distributed computer as (possibly multithreaded) processes. A convenient division is an important factor influencing the resulting performance

Tomas Potuzak

University of West Bohemia, Faculty of Applied Sciences,
Department of Computer Science and Engineering,
Univerzitni 8, 306 14 Plzen, Czech Republic
e-mail: tpotuzak@kiv.zcu.cz

of the entire distributed simulation [3]. During our previous research, we developed a method for road traffic network division, in which a genetic algorithm is employed [4]. The method is sequential and shows good results. However, it is quite slow on a standard desktop computer, especially for large road traffic networks [4].

In this paper, we describe the adaptation of our sequential division method for a distributed/parallel environment in order to maximally utilize the computing power of the environment for a faster division of road traffic networks.

2 Basic Notions

The distributed/parallel method for division of road traffic networks is intended for distributed/parallel road traffic simulation. Such a simulation can be performed as a set of singlethreaded processes on a distributed computer (i.e. without shared memory), a multithreaded process on a parallel computer (i.e. with shared memory), or as a set of multithreaded processes on a distributed/parallel computer (i.e. shared memory among threads of one process, but not among processes) [2].

2.1 *Distributed/Parallel Road Traffic Simulation Description*

The road traffic simulation is most often classified using its level of detail as *macroscopic*, *mesoscopic*, or *microscopic*. The macroscopic simulation deals only with aggregate traffic flows in particular traffic lanes [5]. In the mesoscopic simulation, there is some representation of vehicles (e.g. tasks in queuing networks), but modeling of their interactions is limited [6]. The microscopic simulation considers all vehicles as objects with their own parameters and interactions [7, 8], which makes it very time-consuming and convenient for distributed computing environment [3].

In this case, it is necessary to divide the road traffic network into required number of sub-networks, which are then simulated on particular nodes of the distributed computer as processes. The processes utilize a communication protocol based on message passing for the transfer of vehicles between the sub-networks and also for the synchronization. If the simulation processes are multithreaded, each thread computes part of the road traffic sub-network, which resides in shared memory of the threads. The threads in one simulation process must be synchronized as well [2].

2.2 *Road Traffic Network Division Description*

The division of the road traffic network is necessary only among the simulation processes (one sub-network per process), not among the simulation threads. The simulation threads of one process can access the sub-network in its shared memory and each of them can simulate a part of crossroads, traffic lanes, and so on [2].

Two issues should be considered during the division of road traffic network for the performance reasons – the load-balancing of the resulting sub-networks and the minimization of the inter-process communication. The load-balancing is necessary

in order to achieve similar speeds of the simulation processes and therefore to minimize their mutual waiting [9]. The minimization of the inter-process communication is necessary, since the communication is very slow [9]. The communication can be reduced by minimization of the number of divided traffic lanes, which leads to lower number of transferred vehicles (i.e. lower number of transferred messages, depending on the communication protocol) [3].

2.3 Genetic Algorithms Description

A genetic algorithm is an evolutionary algorithm [10] mimicking the natural genetic evolution and selection in nature in order to solve a problem [11]. Today, the genetic algorithms are widely used for solving of searching and (multi-objective) optimization problems in many domains [12].

A general genetic algorithm works as follows. A representation of an *individual* (usually a vector of binary or integer values) is selected based on the solved problem [13] and a set of individuals – *initial population* – is most often randomly generated. Then, a *fitness value*, representing an objective assessment in relation to the solved problem, is calculated using a *fitness function* for each individual of this initial population [13]. The fitness function can be single- or multi-objective [12]. Once this is completed, a number of individuals with highest fitness values are selected to be “parents” of a new generation. The new generation is then created from these individuals using *crossover* and *mutation* operators. The crossover uses two individuals to produce new individuals (descendants) incorporating information from both parents, which can be then mutated (i.e. partially randomly changed) [10].

The created descendants form a new generation, whose size corresponds to the number of individuals of the initial population. Then, the fitness value is calculated for the individuals of this new generation and the entire process repeats until a preset number of generations is created or a stop condition is fulfilled [14].

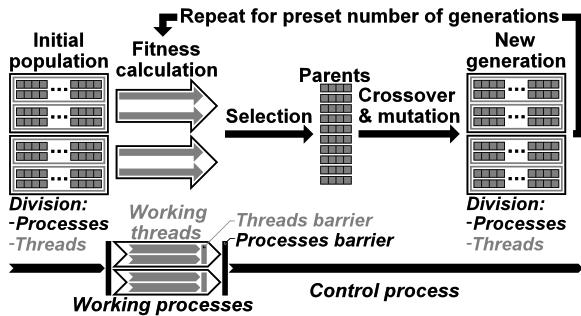
3 Road Traffic Network Division Method

The sequential method for road traffic network division, which we developed, is described in [4] and [14] in detail. The method uses a less-detailed (macroscopic) simulation for assigning of the weights to particular traffic lanes representing the numbers of vehicles in these lanes during the simulation. The traffic network is then considered as a weighted graph with crossroad acting as nodes and sets of lanes connecting the neighboring crossroads acting as edges with weights corresponding to the sum of weights of the particular lanes [4].

3.1 Sequential Dividing Genetic Algorithm Description

The weighted graph is the input for the dividing genetic algorithm (DGA) together with the number of sub-networks, which shall be created. The DGA individual is

Fig. 1 Schema of the D/P-DGA for two processes and two threads per process. The control process performs the division of individuals among working processes, selection, crossover, and mutation. The working processes perform fitness calculation using multiple threads. Barriers are used for synchronization of both threads and processes.



then a vector of integers of length corresponding to the number of crossroads. Each value represents assignment of one crossroad to one sub-network [14].

The initial population consists of 90 randomly generated individuals, for which the fitness values are calculated. The fitness function prefers individuals representing assignment of the crossroads corresponding to load-balanced sub-networks with minimal number of divided traffic lanes between them. So, it consists of two parts – the *equability* representing the load-balancing and the *compactness* representing the minimal number of divided traffic lanes. For more details, see [4] and [14].

Once all fitness values are calculated, ten individuals with highest values are selected for the creation of a new generation using crossover. Each combination of the selected individuals is used to produce two descendants. First descendant receives all integers of even index of the first parent and of odd index of the second parent. Second descendant receives the remainder. Both descendants can be mutated (i.e. random change of several integers). This way a new generation of 90 individuals is created and the entire process repeats for a preset number of generations [4].

3.2 Distributed/Parallel Dividing Genetic Algorithm Description

In order to speed up the DGA, we developed its distributed/parallel version (D/P-DGA), which is the main contribution of this paper. The D/P-DGA is operational, but is still under development. So far, only the fitness calculation is parallelized, since it is the most time-consuming part of the DGA. Moreover, its parallelization is straightforward, because it is an independent computation for each individual.

The implementation of the D/P-DGA is a part of the DUTS Editor, a system for design and division of road traffic networks developed at Department of Computer Science and Engineering of University of West Bohemia (DSCE UWB). The D/P-DGA computation generally consists of multiple multithreaded working processes and one control process, which communicate using message passing. The schema of the D/P-DGA for two processes with two threads per process is depicted in Fig. 1.

4 Tests and Results

The D/P-DGA was thoroughly tested using three computers Intel i7 3.07 GH (4 processor cores with hyper-threading), 12 GB RAM, 1 TB HDD, and Windows 7 interconnected by 1 Gb Ethernet. Three road traffic networks were divided into 4 sub-networks. All three were regular square grids of 64, 256, and 1024 crossroads. The sequential (1 process, 1 thread on 1 computer), the parallel (1 process, 2-4 threads on 1 computer), the distributed (1 control and 2 working processes, 1 thread per working process on 3 computers), and the distributed/parallel (1 control process, 2 working processes, 2-4 threads per working process on 3 computers) executions were tested. The results (averaged from ten attempts) are depicted in Table 1.

Table 1 Results of sequential, parallel, distributed, and distributed/parallel executions

Processes / threads count	1000 generations			10000 generations			100000 generations		
	64	256	1024	64	256	1024	64	256	1024
1 / 1 ^a	618	2443	12689	5601	22826	113632	55385	221702	1105034
1 / 2 ^b	340	1301	6653	3149	12003	58195	30635	119415	576759
1 / 4 ^b	219	741	3585	1945	6726	31787	19076	66767	311794
2 / 1 ^c	713	1781	6669	7061	17167	63267	69934	169770	627977
2 / 2 ^d	608	1253	3968	5842	12002	38812	58227	119026	379333
2 / 4 ^d	539	942	2330	5147	8987	22191	49910	87985	207598

^a Sequential execution ^b Parallel execution ^c Distributed execution ^d Distributed/parallel execution.

The computation time decreases with increasing number of threads. Nevertheless, the results for the distributed/parallel execution are far worse than for the parallel execution, because the individuals and calculated fitness values are transferred between the control and the working processes using the message passing. This significantly degrades the overall performance. The results are better for the largest road traffic network, because there is better computation-to-communication ratio.

5 Conclusion

In this paper, we presented a distributed/parallel dividing genetic algorithm (D/P-DGA) for division of road traffic networks for distributed road traffic simulation. Using the distributed/parallel computation of the fitness values, the total computation time is significantly reduced, which was shown during a thorough testing.

Also, it has been determined that the parallel execution offers better speedup (up to 3.54 using 4 threads) than the distributed/parallel execution (up to 5.32, but using 2 working processes with 4 threads - i.e. 8 threads in total) in comparison

to the sequential execution due to the lack of inter-process communication. Still, the distributed/parallel execution offers good speedup for large road traffic networks and has the advantage of better scalability, since it is possible to add more working processes using larger number of computers.

In our future work, we will focus on better parallelization of the genetic algorithm. So, the selection, crossover, and mutation will be performed in a parallel way similar to the calculation of the fitness values. We will also explore the possibilities of the inter-process communication reduction.

References

1. Nagel, K., Rickert, M.: Parallel Implementation of the TRANSIMS Micro-Simulation. *Parallel Computing* 27(12), 1611–1639 (2001)
2. Potuzak, T.: Distributed-Parallel Road Traffic Simulator for Clusters of Multi-core Computers. In: 2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications – DS-RT 2012, pp. 195–201 (2002)
3. Potuzak, T.: Methods for Reduction of Interprocess Communication in Distributed Simulation of Road Traffic. Doctoral thesis, University of West Bohemia, Pilsen (2009)
4. Potuzak, T.: Methods for Division of Road Traffic Networks Focused on Load-Balancing. *Advances in Computing* 2(4), 42–53 (2012)
5. Lighthill, M.H., Whitman, G.B.: On kinematic waves II: A theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London, S. A* 229, 317–345 (1955)
6. Nizzard, L.: Combining Microscopic and Mesoscopic Traffic Simulators. Raport de stage d’option scientifique Ecole Polytechnique, Paris (2002)
7. Gipps, P.G.: A behavioural car following model for computer simulation. *Transp. Res. Board* 15-B(2), 403–414 (1981)
8. Nagel, K., Schreckenberg, M.: A Cellular Automaton Model for Freeway Traffic. *Journal de Physique I* 2, 2221–2229 (1992)
9. Potuzak, T.: Utilization of a Genetic Algorithm in Division of Road Traffic Network for Distributed Simulation. In: ECBS-EERC 2011 – 2011 Second Eastern European Regional Conference on the Engineering of Computer Based Systems, pp. 151–152 (2011)
10. Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming (2008), Published via <http://lulu.com> and freely available at <http://www.gp-field-guide.org.uk> (with contributions by J.R. Koza)
11. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
12. Farshbaf, M., Feizi-Darakhshi, M.: Multi-objective Optimization of Graph Partitioning using Genetic Algorithms. In: 2009 Third International Conference on Advanced Engineering Computing and Applications in Sciences, pp. 1–6 (2009)
13. Menouar, B.: Genetic Algorithm Encoding Representations for Graph Partitioning Problems. In: 2010 International Conference on Machine and Web Intelligence (ICMWI), pp. 288–291 (2010)
14. Potuzak, T.: Suitability of a Genetic Algorithm for Road Traffic Network Division. In: KDIR 2011 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval, pp. 448–451 (2011)

Environmental Influence in Bio-inspired Game Level Solver Algorithms

Antonio Gonzalez-Pardo and David Camacho

Abstract. Bio-inspired algorithms have been widely used to solve problems in areas like heuristic search, classical optimization, or optimum configuration in complex systems. This paper studies how Genetic Algorithms (GA) and Ant Colony Optimization (ACO) algorithms can be applied to automatically solve levels in the well known Lemmings Game. The main goal of this work is to study the influence that the environment exerts over these algorithms, specially when the goal of the selected game is to save an individual (lemming) that should take into account their environment to improve their possibilities of survival. The experimental evaluations carried out reveals that the performance of the algorithm (i.e. number of paths found) is improve when the algorithm uses a small quantity of information about the environment.

1 Introduction

Bio-inspired algorithm research field has been widely used to solve problems or to search for the optimum configuration of complex systems. Due to these type of problems exhibit *NP-complete* or *NP-hard* complexity, the resolution process needs a huge amount of resources (such as computational effort or time). Some examples of such problems are *scheduling problems*, *constrained satisfaction problems*, or *routing problems*.

A good strategy to reduce the time needed to solve NP-complete problems is applying bio-inspired algorithms, such as evolutionary algorithm (EA) [Fogel, 1995, Eiben and Smith, 2009] or swarm intelligence [Engelbrecht, 2007]. These types of algorithms work with a population of possible solutions that navigates through the solution space of the modelled problem.

Antonio Gonzalez-Pardo · David Camacho
Computer Science Department
Escuela Politécnica Superior
Universidad Autónoma de Madrid
e-mail: {antonio.gonzalez,david.camacho}@uam.es

In the case of EA, each individual is evaluated by a fitness function that allows their comparison. Then, individuals with better fitness value generates the next population by the use of the crossover and mutation [Forrest, 1993]. These operators allow the generation of new individuals taking into account their parents' characteristic.

Swarm intelligence algorithms focus on the collective behaviour of self-organizing systems [Farooq, 2008] where the iterations among individuals generate collective knowledge based on social colonies [Karaboga, 2005]. In this case, the initial population travels through the solution space in order to obtain the best solution to the problem.

In this paper a classical Ant Colony Optimization [Dorigo, 1999] and a Genetic Algorithm are applied to the well-known *Lemmings Game* to determine whether the different levels can be solved using the given skills.

The Lemmings Game is a popular proven NP-hard puzzle game [Cormode, 2004]. In spite of the popularity that this game obtained in the 1990s, few research has been applied to it. Computational intelligence has just been applied to video games such as Mastermind, the Art of Zen, Ms Pac-Man, Tetris or Mario Bros.

This paper tries to determine the influence that environment (in this case, the Lemmings level) exerts over the different algorithms. In order to do that the performance of a Genetic Algorithm (GA), an Ant Colony Optimization (ACO) and an heuristic for the ACO will be analysed.

2 The Lemmings Game

Lemmings are creatures that need to be saved. In each level, Lemmings start in a specific point of the stage and must be guided to the exit point by the player. They live in a two-dimensional space and are affected by gravity. They start walking in a specific direction until they find an obstacle. In this case the Lemming will change the direction and walk back. If the Lemming encounters a hole, it will fall down. The only two ways, considered in this paper, by which a Lemming can die is by falling beyond a certain distance or by falling from the bottom of the level.

In order to make Lemmings to reach the exit point, players have a set of “skills” that must be given (not necessarily all of them) to the Lemmings. Using these skills, Lemmings can modify the environment creating tunnels, or bridges, and thus creating a new way to reach the exit. There are eight different skills that allow Lemming to have an umbrella, to dig or to climb, amongst others. Each skill can be used (i.e. assigned) a maximum number of times. It is not necessary to use all of the skills in the levels.

In the Lemmings' world, there are a huge number of materials, but all of them can be grouped in two different classes: the ones that can be modified (e.g. can be dug) and the ones that cannot be altered. In the case that a Lemming is digging and finds a material that cannot be dug, the Lemming will stop digging and start walking.

3 Description of the Studied Algorithms

In this paper GA and ACO algorithms are studied. The aim of the experimental phase is to determine whether these algorithms can find the different paths that guide Lemmings to the exit point of the level.

3.1 Genetic Algorithm

The GA applied in this work, initializes individuals with a random phenotype length. The maximum length of the phenotype depends on the maximum time of the level or the maximum genotype length allowed. The phenotype is a list of genes where each gene (T, S) contains the skill (S) that is going to be executed in the step T . Both values (the step and the skill) are selected randomly depending on the maximum time given to solve the level, and the total number of remaining skills. The phenotype represents the different decisions that the player could make. This phenotype is then evaluated against the level. The lemming starts its execution applying the skills specified in the given steps.

The goal of the GA is to maximize the fitness function represented by Eq. 4 and it is composed by Eq. 1, Eq. 2 and Eq. 3. Eq. 1 takes into account the time spent by the Lemming to solve the level. Eq. 2 is used to favour those paths that use less actions, or less skills. Finally, another key concept is the number of lemmings saved in the level represented in Eq. 3.

$$T(Ind_i) = \text{MaxTime} - \text{Time}(Ind_i) \quad (1)$$

$$A(Ind_i) = \text{TotalActGiv} - \text{ActionUsed}(Ind_i) \quad (2)$$

$$S(Ind_i) = \text{TotalLemm} - \text{BlockersUsed}(Ind_i) - \text{ExplodedLemming}(Ind_i) \quad (3)$$

$$F(Ind_i) = \frac{T(Ind_i) + A(Ind_i) + S(Ind_i)}{\text{MaxTime} + \text{TotalActGiv} + \text{TotalLemm}} \quad (4)$$

Although ACO will use the same function to evaluate the goodness of the paths, only GA can produce negative fitness. This negative fitness value is obtained if the individual produces an invalid path (i.e. in the evaluation, the lemming is not able to reach the exit point or the lemming dies trying it). In this case the fitness value is $-1 * F(Ind_i)$.

3.2 Ant Colony Optimization

In this work, the ants are the Lemmings of the game. So, they start in the entry point of the level and at each position, each ant will decide whether to continue executing the current action or to select another skill to execute. This decision depends on the number of remaining skills and the pheromones deposited in the current location. This means that skills with a higher pheromone value, and/or that can be applied several times, have more chances to be selected. Once the ant decides to change its

behaviour, it deposits a pheromone in the current location and continues with its execution.

A pheromone is an object that represents a decision taken previously by any ant. It means that some ant had taken a specific skill at this point. For that reason each pheromone contains the name of the skill taken, the direction in which this skill had been executed and a value representing the goodness for this decision (this value is computed at the end of the ant execution).

In this work when any ant reaches the exit of the level, it will start again from the entry point updating all the pheromones with the fitness value of the path. This fitness is the same as the one used by the GA (Eq. 4). Finally, when all pheromones have been updated and the ant arrives to the destination (for the second time), it will forget the followed path and it will start again as a new ant, without any knowledge about the location of the exit.

3.3 *The Common-Sense Ant*

The problem with the described ACO algorithm is that randomness may generate strange situation like having an ant falling down and suddenly the ants start building a bridge in the air. For this reason, a common-sense heuristic is defined.

With this heuristic, the current skill applied and the ants' environment will influence in the decision of the next skill to apply. This heuristic will avoid the use of a skill when it cannot be executed, for example, it is not useful to apply the Climber skill if the ant is not in front of a wall.

4 Experimental Phase

The aim of the experiments is to analyse the influence of the environment in the behaviour of a GA and an ACO algorithms. Five different levels have been designed, the easiest one and the hardest one are shown in Fig. 1(a) and Fig. 1(b), respectively. This valuation is based on the size of the level, the different blocks contained into each level, the distance from the entry point to the exit point, the number of skills needed to solve the level, etc. It is important to take into account that all the terrains used in the levels are editable terrain, i. e. ants can dig, climb and bash into it.

For each level, GA, ACO and ACO with common-sense ants are executed. Each execution is repeated 50 times because all algorithms are stochastic. In order to compare the results obtained among the different algorithms there are some parameters that are common to all of them. Each experiment uses the Eq. 4 to evaluate the different paths, the algorithms execute 100 ants (or individuals) during 200 iterations(or generations), and each experiment is repeated 50 times.

The GAs have, also, the following configuration: the maximum phenotype length is 20, the probability of having One-point crossover is 90%, while the mutation rate is fixed to 10%. There is not elitism and the goal is to maximize the results obtained from the fitness function.

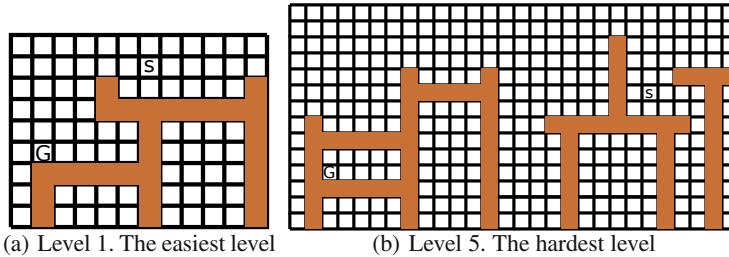


Fig. 1 Examples of the easiest and the hardest level modelled in this work

The goal of the experiments is to compare the number of different solutions that each algorithm is able to build. This information is shown in Table 1. The values correspond to the number of different paths found in 50 executions of the algorithms.

Table 1 Number of different solutions found by the algorithms described

Level	# Different Solutions (Genetic Algorithm)	# Different Solutions (ACO Random Ant)	# Different Solutions (ACO Common-Sense)
1	3219	3868	2516
2	12629	4463	4042
3	370	1130	2487
4	2	15	32
5	0	3	7

5 Conclusions

This paper analyses the possibility of applying Genetic Algorithm and Ant Colony Optimization to generate automatic game level solver tools. The application domain of this work is the well-known Lemmings game, where Lemmings have to apply different skills in order to reach the exit. Five different levels have been designed with different complexity depending on the size of the level, the number of available skills, or the distance between the start and the destination point, amongst others.

Experimental results reveal that both algorithms can successfully be applied to solve the levels. Nevertheless, as it can be seen in Table 1 the Genetic Algorithms provide less different paths when the levels are harder (i.e. levels 4 and 5). This is produced because GA generates individuals without taking into account the level landscape (i.e. it is a blind generation of individuals). On the other hand, ACO uses the terrain information to apply different skills at specific steps and thus, it provides better results.

One of the problem observed in this work is that GA does not guarantee the goodness of the following generation of individuals. Parents are selected depending on its fitness value (better fitness value implies more chances for being selected), but crossover and mutation operations do no guarantee that the generated children have

better fitness values. Ant Colony Optimization does not have this problem, because ants are guided by the pheromone trails and then good decision (represented in pheromones with high values) will have higher probabilities for being selected.

In order to determine whether the environment is important for generating automatic level solvers, an heuristic for ACO has been designed. Although this heuristic (called *common-sense heuristic*) provides less different paths in easiest levels (level 1 and 2), it provides better results when it tries to solved the hardest levels (levels 3, 4 and 5). This fact demonstrates that level information is very important to make automatic level solvers.

Nevertheless, the design of domain based heuristics makes the platform very dependent on the domain and thus the conclusions, and the approach, are not applicable to other different domains.

Acknowledgements. This work has been supported by the Spanish Ministry of Science and Innovation under grant TIN2010-19872.

References

- [Cormode, 2004] Cormode, G.: The hardness of the lemmings game, or oh no, more npcompleteness proofs. In: Proceedings of Third International Conference on Fun with Algorithms, pp. 65–76 (2004)
- [Dorigo, 1999] Dorigo, M.: Ant colony optimization: A new meta-heuristic. In: Proceedings of the Congress on Evolutionary Computation, pp. 1470–1477. IEEE Press (1999)
- [Eiben and Smith, 2009] Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer (2009)
- [Engelbrecht, 2007] Engelbrecht, A.: Computational Intelligence: An Introduction, 2nd edn. Wiley Publishing (2007)
- [Farooq, 2008] Farooq, M.: Bee-Inspired Protocol Engineering: From Nature to Networks. Springer Publishing Company, Incorporated (2008)
- [Fogel, 1995] Fogel, D.B.: Evolutionary computation: toward a new philosophy of machine intelligence. IEEE Press (1995)
- [Forrest, 1993] Forrest, S.: Genetic algorithms: principles of natural selection applied to computation. *Science* 261(5123), 872–878 (1993)
- [Karaboga, 2005] Karaboga, D.: An idea based on honey bee swarm for numerical optimization. *Techn. Rep. TR06*, vol. 129(2), p. 2865. Erciyes Univ. Press, Erciyes (2005)

The Impact of the “Nogood Processor” Technique in Scale-Free Networks

Ionel Muscalagiu, Horia Emil Popa, and Viorel Negru

Abstract. DisCSPs are composed of agents that manage variables which are connected by constraints, various algorithms for solving DisCSPs are searching through this network of constraints. The scale-free graphs have been proposed as a generic and universal model of network topologies that exhibit power-law distributions in the connectivity of network nodes. Little research was done concerning the network structure for DisCSP and in particular for scale-free networks. The asynchronous searching techniques are characterized by the occurrence of the nogood values during the search for the solution. In this article we analyzed the distribution of nogood values to agents and the way to use the information stored in the nogood, what we will call the nogood processor technique. We examine the effect of nogood processor for networks that have a scale-free structure. We develop a novel way for the distribution of nogood values to agents, the experiments show that it is more effective for several families of asynchronous techniques.

1 Introduction

Constraint programming is a programming approach used to describe and solve large classes of problems such as searching, combinatorial and planning problems. A Distributed Constraint Satisfaction Problem (DisCSP) is a constraint satisfaction problem in which variables and constraints are distributed among multiple agents. This type of distributed modelling appeared naturally for many problems for which the information was distributed to many agents. The idea of sharing various parts

Ionel Muscalagiu
The “Politehnica” University of Timisoara,
The Faculty of Engineering of Hunedoara, Revolutiei, 5, Romania
e-mail: ionel.muscalagiu@fih.upt.ro

Horia Emil Popa · Viorel Negru
The University of the West,
The Faculty of Mathematics and Informatics, Timisoara, V. Parvan 4, Romania
e-mail: {hpopa, vnegru}@info.uvt.ro

of the problem among agents that act independently and collaborate in order to find a solution by using messages has led to the formal problem known as the Distributed Constraint Satisfaction Problem (DisCSP) [10], [6]. DisCSPs are composed of agents, each owning its local constraint network. Variables in different agents are connected by constraints forming a network of constraints. Agents must assign values to their variables so that all constraints between agents are satisfied. Distributed networks of constraints have proven their success in modeling real problems.

There are complete asynchronous searching techniques for solving the DisCSP in this constraints network, such as the ABT (Asynchronous Backtracking), AWCS (Asynchronous Weak Commitment), ABTDO (Dynamic Ordering for Asynchronous Backtracking) [6],[10].

In recent years various complex networks have been identified as having a scale-free structure [3], [2]. Scale-free networks are abundant in nature and society, describing such diverse systems as the Internet, a network of routers connected by various physical connections, the chemical network of a cell,etc. Little research was done concerning the behaviour of the search techniques in networks of constraints that have a structure of the scale-free networks type [9]. Thus, few things are known about choosing the optimal search technique for topological structures of the scale-free network type. The purpose of the article is to develop search algorithms specialized for scale-free networks of constraints, algorithms that require minimum costs for obtaining the solution.

In the distributed constraint satisfaction area, the asynchronous weak-commitment search algorithm (AWCS) [10], plays a fundamental and pioneer role among algorithms for solving the distributed CSPs. The algorithm is characterized by an explosion of the nogood values, but, by dynamically changing the agents' order, an efficient algorithm is obtained.

The occurrence of nogood values has the effect of inducing new constraints. Nogood values show the cause of failure and their incorporation as a new constraint will teach the agents not to repeat the same mistake. The non - restriction for recording the nogood values could become, in certain cases, impracticable. The main reason is that the storing of nogood values excessively consumes memory and could lead to lowering the memory that has been left. Another unpleasant effect of storing a large number of nogood values is related to the fact that the verification of the current associations in the list of nogood values that are stored becomes very expensive, the searching effort removing the benefits brought by the nogood values storing. These elements are analyzed as targeting to see if this nogood processor technique brings benefits in terms of efficiency.

In [1] is introduced for the first time the notion of nogood processor. In [8] we try to adapt the nogood processor technique for the AWCS technique. This technique consists in storing the nogood values and further use the information given by nogoods in the process of selecting a new value for the variables associated to agents.

In this paper we examine the effect of nogood processor for constraints networks of the scale-free type. We develop a novel way for the distribution of nogood values to agents, the experiments show that it is more effective for several families of

asynchronous techniques. Starting with the results from [8], this study tries to adapt the version of nogood processor with the learning techniques in the purpose of finding a solution that increases the performances of the AWCS technique. The aim of these studies is to develop search algorithms specialised for scale-free networks.

The evaluation of the performances of the AWCS technique is done using NetLogo. NetLogo is a programming environment with agents that allows the implementation of the asynchronous techniques ([11], [12]). In order to make such estimation, the AWCS technique with nogood processor are implemented in NetLogo, using the models proposed in [7], model calling DisCSP-NetLogo. Implementation examples for the AWCS family can be found on the website [12]. For the first time, the experiments are done with a large number of agents (500 and 1000), using a NetLogo model that runs on a cluster.

2 The Framework

This paragraph presents some notions related to the DisCSP modeling and AWCS algorithm [10], [6].

Definition 1. 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 2. 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 $\phi : X \longrightarrow A$ is a function that maps each variable to its agent.

In this article we will consider that each agent A_i has allocated a single variable x_i , thus $p = n$. Also, we assume the following communication model [10]:

- agents communicate by sending messages. An agent can send messages to other agents iff the agent knows the addresses of the agents.
- the delay in delivering a message is finite, although random. For transmission between any pair of agents, messages are received in the order in which they were sent.

Asynchronous search algorithms are characterized by the agents using the messages during the process of searching the solution. Typically, it uses two types of messages:

- the *ok* message, which contains an assignment variable-value and is sent by an agent to the constraint-evaluating-agent in order to see if the value is right.

- the *nogood* message, which contains a list (called nogood) with the assignments wherefore a looseness was found, is sent in case the constraint-evaluating-agent finds an unfulfilled constraint.

Definition 3. Two agents are connected if there is a constraint among the variables associated to them. Agent A_i has a higher priority than agent A_j if A_i appears before A_j in the total ordering. Agent A_i is the value-sending agent and agent A_j the constraint-evaluating agent.

Definition 4. The *agent – view* list belonging to an agent A_i is the set of the newest associations received by the agent for the variables of the agents to whom it's connected.

Definition 5. The *nogood* list is a set of associations for distinct variables for which an inconsistency was found (an unsatisfied constraint). The *agent – view* list together with the stored *nogood* values constitutes the working context of each agent, depending on them the agent makes decisions.

Definition 6. A *nogood* list received by agent A_i is consistent for that agent, if it contains the same associations as *agent – view* for all the variables of the parent agents A_k connected with A_i .

The AWCS algorithm [10] is a hybrid algorithm obtained by the combination of ABT algorithm with WCS algorithm, which exists for CSP. It can be considered to be an improved ABT variant, but not necessarily by reducing the nogood values, but by changing the priority order. It deliberately follows to record all the nogood values to ensure the completeness of the algorithm, but also to avoid some unstable situations.

The authors show in [10] that this new algorithm can be built by a dynamical change of the priority order. The AWCS algorithm uses, like ABT, the two types of ok and nogood messages, with the same significance. There is a major difference in the way it treats the ok message. In case of receiving the ok message, if the agent can't find a value to its variable that should be consistent with the values of the variables that have a greater priority, the agent not only creates and sends the nogood message, but also increases the priority in order to be maximum among the neighbors.

3 Scale-Free Network

The study of complex network topologies across many fields of science and technology has become a rapidly advancing area of research in the last few years. One of the key areas of research is understanding the network properties that are optimized by specific network architectures. The last few years have led to a series of discoveries that uncovered statistical properties that are common to a variety of diverse real-world social, information, biological, and technological networks

In recent years various complex networks have been identified as having a scale-free structure [3], [2]. Scale-free networks are abundant in nature and society, describing such diverse systems as the Internet, a network of routers connected by various physical connections, SNS, the chemical network of a cell.

Not all nodes in a network have the same number of edges. The spread in the node degrees is characterized by a distribution function $P(k)$, which gives the probability that a randomly selected node has exactly k edges. Since in a random graph the edges are placed randomly, the majority of nodes have approximately the same degree. One of the most interesting developments in our understanding of complex networks was the discovery that for most large networks the degree distribution significantly deviates from a Poisson distribution. In particular, for a large number of networks, including the World Wide Web, the Internet or metabolic networks the degree distribution follows a power-law for a large number of nodes [3], [2]. Such networks are called scale free [2].

A scale-free network is characterized by a power-law degree distribution as follows:

$$p(k) \propto k^{-\gamma} \quad (1)$$

where k is the degree and γ is the exponent that depends on each network structure. Scale-free networks have no scale because there is no typical number of links [2]. The random network models assume that the probability that two nodes are connected is random and uniform. In contrast, most real networks exhibit some preferential connectivity.

4 Nogood Processor in Scale-Free Network

A DisCSP can be represented by a constraint graph $G = (X, E)$, whose nodes represent the variables and edges represent the constraints:

1. $X = \{x_1, \dots, x_n\}$ is a set of n nodes/variables
2. $E = \{(x_i, x_j)\}$ is a set of edges for which we have a constraint between variables x_i and x_j .

In the scale-free networks some nodes concentrate very much connections, the rest of the nodes having very few connections. Starting from this observation we define the notion of hub:

Definition 7. A node is called a hub if it has a larger number of connections than a constant $c \in N$. Let H be set of hubs $H = \{x_i | x_i \in N, \deg(x_i) \geq c\}$ where $\deg(x_i)$ is the degree of node x_i .

Let us show a simple example. A constraint network of a DisCSP having a scale-free network structure, with 25 nodes and the minimal degree 2, is represented in fig.1(a). If we consider the constant $c = 8$, we can remark the existence of a number of 4 hubs: $H_1=2$ (degree 12), $H_2=0$ (degree 10), $H_3=1$ (degree 9) and $H_4=3$ (degree 8). For nodes H_1, H_2, H_3, H_4 we can remark that they have very many connections compared with the rest, that have very few connections (see also fig.1(b)).

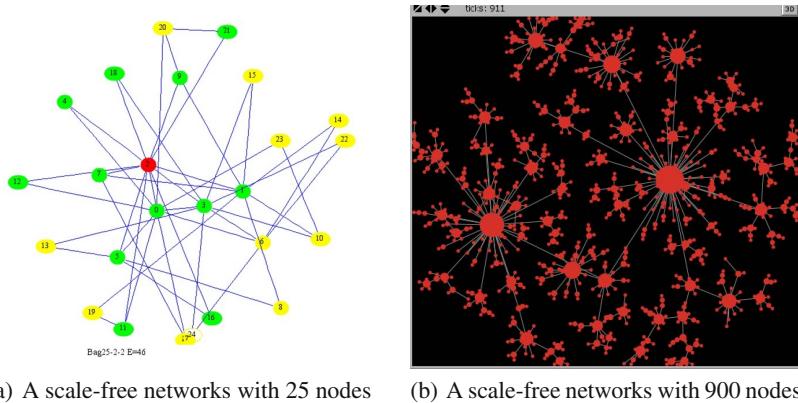


Fig. 1 Constraint network representing a DisCSP where have a scale-free networks

In the AWCS algorithm when a nogood message is received by the agent A_i , the agent adds the nogood value to its nogood store and executes a verification of the inconsistencies for nogood. If the new nogood has also an unknown variable, the agent needs to receive from the corresponding agent the value of the variable that's been cared for. Unfortunately, the information from nogood is not completely used in the case of attributing a new value for the variable associated to the agent. It is possible that the nogood values contain a reference to this value, that implying the attribution has appeared before as inconsistent. The use of this information will be the basis of the nogood processor technique construction.

In this paper it is considered that some agents have access to the results of their own nogood processor. More, each agent sends (stores) nogood values that it has received to the associated nogood processor. The information stored by each nogood processor will be used in searching a new value for each variable cared for by the agent. For this, each nogood processor will verify (asked by an agent) through its subroutine check-inconsistent-value-nogood-processor, if the value selected by the agent had no previous existence associated with the higher priority agents values. In fig. 2 we have the checking routine for the inconsistency of a new value [8].

Another question needing an answer was identifying how the nogood processors are distributed. Practically, each agent, when receiving a nogood, sends this to a associated nogood processor that stores it into a nogood-store list (nogood processor only saves those new nogood values, eliminating the copies). The information will be used later when searching for a new value. Therefore the check-agent-view procedure will select a new value consistent with the agent view list and with the nogood list stored by the nogood processor (the value will be selected if, complementarily, the check-inconsistent-value-nogood-processor subroutine will return the consistent value).

The check-inconsistent-value-nogood-processor routine is used just for the higher priority agents, priority considered at the moment of nogood value storing. To put it

```

function check-inconsistent-value-nogood-processor [ $A_i$ ]
    foreach Nogood ∈ nogoods-store do
        foreach x ∈ Nogood with the current priority from current-view * bigger than the agent's  $A_i$ 
            pos ← position x in Nogood
            if x != item pos current-value
                return consistent
            endif
        end do
        if current-value != item  $A_i$  in nogood
            return consistent
        endif
    end do
    return inconsistent
end procedure

```

Fig. 2 The Procedure check-inconsistent-value-nogood-processor

differently, the identification of the agents with higher priority towards agent A_i , is not made by using their actual priority (in current-view), but the priority stored by the nogood processor.

5 Experimental Results

In this paragraph we will present our experimental results, obtained by implementing and evaluating the asynchronous techniques AWCS. The families of AWCS techniques are implemented in NetLogo [11], [12]. The implementation and evaluation is done using the two models proposed in [7]. The Netlogo implementations were run on a cluster of computers using RedHat Linux. The cluster allowed running instances of 500 and 1000 agents, with various difficulties.

In this paper, the Java program developed by Sun Microsystems Laboratories is used as a scale-free network formation tool [4]. This program can generate scale-free networks giving the number of nodes, the minimal degree of each agent md. Scale-free networks are generated by the tool with the following parameters:

- nodes = 500, md = 4 and $\gamma = 1.8$
- nodes = 1000, md = 2 and $\gamma = 2.1$.

We examine the performance of AWCS in scale-free networks. Specifically, we implemented and generated in NetLogo both solvable and unsolvable problems that have a structure of scale-free networks. Implementation examples for the scale-free network instance generator (using scale-free networks from [4]) can be found on the website [12]. We set the domain size of each variable to ten, i.e., domain = 10 which means $|D_i| = 10$. For the evaluations, we generate five scale-free networks. For each network, the constraint tightness is varied from 0.1 to 0.9 by 0.1. For each constraint tightness, 200 random problem instances are generated. Thus, the results represent the averages of these 1000 instances for each of the five networks. These problems have a number of variables with a fixed domain. This creates problems that cover

a wide range of difficulty, from easy problem instances to hard instances (for each version we retaining the average of the measured values).

In order to evaluate the asynchronous search techniques, the message flow was counted i.e. the quantity of ok and nogood messages exchanged by the agents, the number of checked constraints i.e. the local effort made by each agent, and the number of nonconcurrent constraints checks (defined in [6], noted with ncccs) necessary for obtaining the solution.

Asynchronous techniques use some message processing routines. In this paper we analyze implementations of the AWCS family with complete processing of messages: each agent treats entirely the existing messages in its message queue. In fig. 3 shows the new handle-message procedure.

```

to handle-message [msize]
  set nrm 0
  1 while [not empty? message-queue and nrm <= mszie] or
  1' while [not empty? message-queue] ***
  [
    set msg retrieve-message
    ...
    set nrm nrm + 1
  ]
  If (Nrm !=0 )
    [Check-agent-view]
  end

```

Fig. 3 The handle-message for the AWCS

Concerning the complete or partial processing of the messages, it can be done by means of the mszie variable or renouncing of the second condition of package limitation. That has a role to decide the number of extracted and processed messages from the message queue. If mszie is equal to the number of elements from the message queue ($\text{mszie}=\text{length}(\text{message-queue})$) or if that condition is missing, the procedure in fig. 3 allows the processing of all the messages. The first variant supposes the insertion of line 1 instead of line 1'. In that case, each agent stops in the moment in which either it has no more messages or mszie messages were processed. The second variant supposes the insertion of line 1'. Message processing supposes an effort and thus the occurrence of a delay. It is possible that other messages arrive beside those from the initial moment. In the case of the second variant if later new messages appear, those are still treated thus surpassing the number mszie of messages allowed. In this paper we implement both variants.

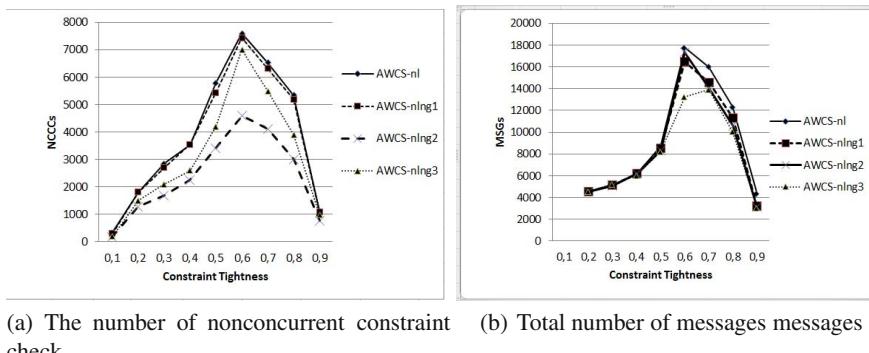
In the AWCS family there are many variants that are based on building of efficient nogoods (nogood learning [5]) or on storing and using those nogoods in the process of selecting the values (nogood processor). Two families of AWCS techniques are evaluated:

- the basic variant proposed in [10] improved with the nogood learning technique (noted with AWCS-nl).
- the basic variant proposed in [10] improved with the nogood learning technique and the nogood processor technique (noted with AWCS-nlng).

Four implementations are done corresponding to the obtained models:

- the basic variant proposed in [10] improved with the nogood learning technique (noted with AWCS-nl).
- variant based on the nogood processor distributed to each agent: AWCS-nlng₁.
- variant based on the nogood processor distributed only to the hub type agents: AWCS-nlng₂. This variant used the first method of message processing
- variant based on the nogood processor distributed only to the agents of the hub type: AWCS-nlng₃. This variant used the second solution of message processing.

The number of concurrent constraint checks (ncccs) allows the evaluation of global effort without considering that the agents work concurrently (informally, the number of concurrent constraint checks approximates the longest sequence of constraint checks not performed concurrently). Analyzing the results from fig 4(a), one can remark that the method that distributes the nogood processors to the hub type nodes reduces the global effort made by the agents (AWCS-nlng₂ and AWCS-nlng₃). According to definition 7, the number of hubs depends on the constant c . For the problems of the scale-free network type evaluated in this article, we considered as hub type nodes those having the degree over 60% of the maximum degree (approximately $c=3$ for nodes=500 and $c=4$ for nodes=1000).



(a) The number of nonconcurrent constraint check (b) Total number of messages messages

Fig. 4 Comparative study for the AWCS versions-(Scale-free Networks)

In case of instances with 1000 agents, the four implementation had the same behaviour, the version AWCS-nlng₃ requiring the least effort for obtaining the solution.

In the case of the message flow (fig 4(b)), one can notice almost equal efforts for obtaining the solution for all the 4 versions. Though, for problems with high difficulty (constraint tightness = 0.6) the AWCS-nlng₃ variant requires slightly less messages.

Regarding the complete processing of the messages, experimental analysis shows that the first solution that processes completely the messages (AWCS-nlg₂) is better. It remains to experiment, in the future, for a higher number of agents (10000) and for other instances of scale-free networks (md=16, md=32) if the behaviour of the three techniques remains the same.

6 Conclusions

In this paper we examine the effect of nogood processor for constraints networks of the scale-free type. We develop a novel way for the distribution of nogood values to agents, the experiments show that it is more effective for several families of asynchronous techniques.

We analyzed more versions obtained by distributing the nogood values to more nogood processors for each agent for constraints networks of the scale-free type.

The experimental analysis shows that the best way of distribution is that in which the nogood processors are distributed to the nodes of the hub type (that have a very high degree, compared to other nodes). That variant requires the lowest costs.

The most performant version was obtained by distributing the nogood values only to the agent of the hub type in the case of treating all the messages from the queue, including those that had appeared later.

We believe that the combination proposed in this article could bring important benefits to the performances of the asynchronous techniques, leading to the reduction of the effort in finding the solution in the case of constraints networks of the scale-free type.

References

1. Armstrong, A., Durfee, E.: Dynamic Prioritization of Complex Agents in Distributed Constraint Satisfaction Problems. In: Proceedings of the 15th IJCAI, Nagoya, Japan, pp. 620–625 (1997)
2. Barabasi, A.L., Albert, A.L.: Emergence of scaling in random networks. *Science* 286, 509–512 (1999)
3. Albert, R., Barabási, A.-L.: Statistical mechanics of complex networks. *Rev. Mod. Phys.* 74, 47–97 (2002)
4. Densmore, O.: An exploration of power-law networks (2009), <http://backspaces.net/sun/PLaw/index.html>
5. Hirayama, K., Yokoo: The Effect of Nogood Learning in Distributed Constraint Satisfaction. In: Proceedings of the 20th IEEE International Conference on Distributed Computing Systems (ICDCS 2000), pp. 169–177 (2000)
6. Meisels, A.: Distributed Search by Constrained Agents: algorithms, performance, communication, pp. 105–120. Springer, London (2008)
7. Muscalagiu, I., Jiang, H., Popa, H.E.: Implementation and evaluation model for the asynchronous techniques: from a synchronously distributed system to a asynchronous distributed system. In: Proceedings of the 8th SYNASC Conference, Timisoara, pp. 209–216 (2006)

8. Muscalagiu, I., Cretu, V.: Improving the Performances of Asynchronous Algorithms by Combining the Nogood Processors with the Nogood Learning Techniques. Journal "INFORMATICA" 17(1) (2006)
9. Okimoto, T., Iwasaki, A., Yokoo, M.: Effect of DisCSP variable-ordering heuristics in scale-free networks. Multiagent and Grid Systems 8, 127–141 (2012)
10. 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)
11. Wilensky, U.: NetLogo itself:NetLogo. Center for Connected Learning and Computer-Based Modeling, Evanston (1999),
<http://ccl.northwestern.edu/netlogo/>
12. MAS NetLogo Models-a, <http://discsp-netlogo.fih.upt.ro/>

Knowledge-Based Agent for Efficient Allocation of Distributed Resources

Ebrahim Nageba, Mahmoud Barhamgi, and Jocelyne Fayn

Abstract. The mobilization of heterogeneous distributed resources and the allocation of resources to user-tasks are still challenges in distributed computing. In this paper, we propose a knowledge-based agent to solve the problem of resources mobilization and allocation taking into account the continuous changing of resources conditions. The agent model we propose is mainly based on ontological models representing basic collaborative environment entities, rules, inference engines, and object oriented components for resources data processing. We suggest mechanisms to handle the discovered resources in terms of updating, filtering, ranking, and allocation.

Keywords: Ontology, Resources management, Task-based computing.

1 Introduction

In a collaborative environment, different actors participate in the realization of various tasks in a specific domain. These actors belong to different organizations which are involved in one or more business networks. Additionally, these actors have at their disposal high-tech communication devices, e.g., Smartphones, Tablet PCs, etc, to access information anytime, anywhere, and to exchange huge amounts of data. Normally, users' tasks achievement requires different types of resources which are semantically heterogeneous and managed by numerous business partners or organizations. The aforementioned context enables powerful collaboration among the different organizations being involved in business processes. The more the number

Ebrahim Nageba · Mahmoud Barhamgi

Université Lyon 1, 69622 Villeurbanne, France

e-mail: {ebrahim.nageba, mahmoud.barhamgi}@univ-lyon1.fr

Jocelyne Fayn

SFR Santé Lyon Est, Université Lyon 1, 69677 Bron, France

e-mail: jocelyne.fayn@insa-lyon.fr

of organizations involved in business processes increases, the more the volume of the used resources becomes huge and the more the conditions of the resources managed by the involved organizations are subject to continuous changing. For example, some services may be unavailable, data sources become inaccessible, devices are momentarily broken down or not available, etc. In other words, intensive internet based collaboration makes resources discovery, mobilization, and allocation a big challenge due to the dynamic context of heterogeneous and distributed resources. Resources availability, quantities, and accessibility, as well as other QoS parameters can be considered as being resources meta-data. Unfortunately, organizations lack well defined policies and tools to provide other business partners with meta-data of their resources and do not pay enough attention to actualize resources meta-data. To empower inter-organizational collaboration, resources meta-data must be up to date and available in real time. Thus, one of the major challenges within collaborative environments is the continuous changing of resources conditions which limits the capabilities of information systems to react to resources demand and to achieve their tasks in the attributed time. To overcome this challenge, we shall create a mechanism permitting to automatically mobilize and allocate only the required resources meeting some QoS criterions, i.e. availability, accessibility, execution time, etc. In this paper, we suggest a Resource Mobilization Agent (RMA) based on: (i) ontological models representing the different entities of collaborative environments, i.e. Actor, Task, Object, Resource, Organization, Process, Service, Location, Parameter, ... as well as the associations among these entities; (ii) A model of rules including a set of statements that specify how to link each task with the required resources, and an inference engine to perform rule-based reasoning; (iii) Object oriented components to handle heterogeneous resources data in terms of mapping, updating, filtering, and ranking. The aim of our the proposed RMA is to automatically allocate to the user-tasks the resources discovered in a dynamic context.

This paper is organized as follows. Section 2 presents the architecture model of RMA and its main components. Section 3 applies our model on an example from the Healthcare domain. Section 4 reports related research works. In section 5 we conclude this paper.

2 RMA Architecture

We propose a hybrid architecture of RMA which enables the allocation of heterogeneous and distributed resources to each user-task, depending on the status of the object treated by the user-task. Additionally, RMA architecture takes into account the continuous changes of the resources conditions. Figure 1 shows a global view of the RMA architecture which is mainly based on: (i) ontological models constituting a knowledge layer of all types of resources, i.e. services, devices, equipments, etc.; (iii) an inference engine executing a rule-based reasoning in order to deduce the resources required by the user-tasks; (iv) object-oriented components for resources data processing. The knowledge base is the key component of the RMA architecture. It is based on a knowledge meta-model consisting of a collection of generic and

domain-independent ontological models and of the associations which connect these models. The objective of the ontological models is to represent, store, and structure the knowledge of different entities of collaborative environments. The meta-model structure is defined by combining different common physical and abstract entities of the collaborative environments such as Actor, Task, Object, Process, Resource, Organization, Location, etc. (Figure 2). Each of these entities will be described by an ontology including classes, subclasses, object type properties, data type properties, and restrictions. For instance, the Actor ontological model represents the entities that can interact with an information system and access the data. Generally, an Actor belongs to an Organization and he can perform different tasks according to his profile. It is necessary to specify for each actor profile the different tasks that he is authorized to perform. We have therefore defined restrictions to decide if the task is or is not accessible by an actor according to the type of class to which the actor belongs.

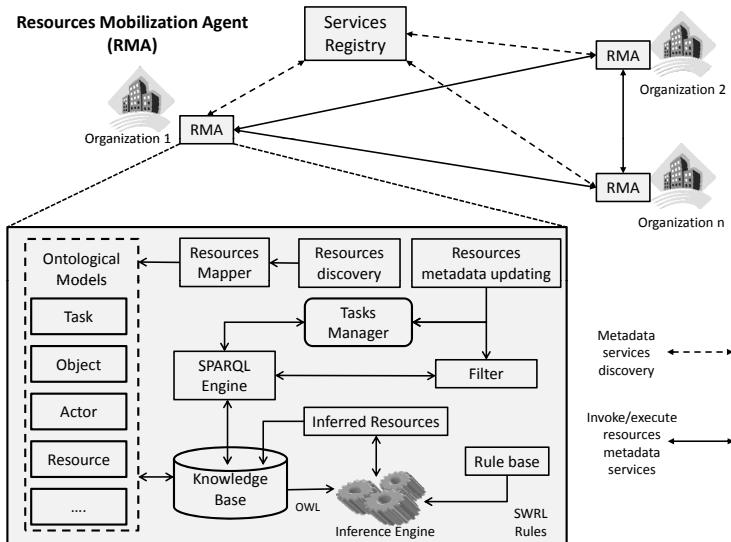


Fig. 1 Resources Mobilization Agent (RMA) architecture

3 Model Evaluation

A typical scenario can be, for example, a Patient Transfer from an accident site to a hospital or from one hospital to another one having additional capabilities for patient treatment. In fact, the Patient Transfer scenario becomes a complex one in some particular contexts such as the accident location hostility, strong traffic density in big cities, roads blocked due to bad weather, the need of an helicopter to evacuate the victims, etc. In a Patient Transfer scenario, the rescue team members or

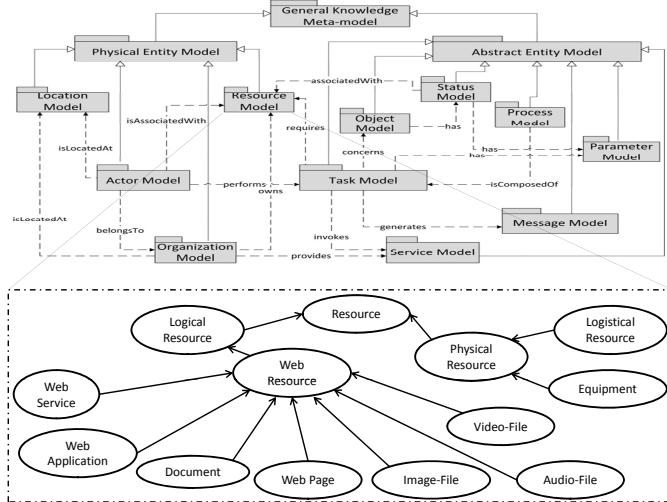


Fig. 2 Graphical representation of our knowledge meta-model using UML notation. The resource ontological model has been selected as example.

the persons who should provide a primary medical assistance to the victim need to know where to transfer him. In addition to the above mentioned environment conditions, we must also consider other factors related to the clinical status of the casualty and also the nearest hospitals abilities, in terms of accommodation, competence and availability of different required resources, to receive the patient. Thus, it is relevant to use the proposed RMA to provide solutions to the healthcare professional according to the requirements mentioned above where the RMA shall perform rule-based reasoning on the facts related to the aforementioned contextual factors, i.e. patient conditions, resources, and environment. The results of the reasoning process will be contextual knowledge about the material resources, including logistics and medical equipments, which are required by the task being performed. To mobilize and allocate the heterogeneous distributed resources needed by the task "Patient Transfer", we propose to instantiate the RMA knowledge meta-model in the eHealth domain. To simulate the resources mobilization and allocation process, we have used Protege, a knowledge acquisition tool that facilitates ontology creation, editing, encoding and management through a friendly graphical user interface. The DL extension provides OWL with a rich set of primitives, i.e., intersection ?, union ?, complement, etc. It allows the ontology designer to define restrictions and conditions on the ontology classes. It is therefore possible to automatically determine the classification hierarchy and check for inconsistencies in any ontology that conforms to OWL-DL. We have employed the Pellet Reasoner to test the ontology consistency and class hierarchy. The inference engine JESS has been used to apply the SWRL rules that we have defined to infer the resources required by the user-tasks. In a typical

Patient Transfer scenario, 89 concepts and 29 axioms are exported in average to the JESS inference engine, and the number of inferred axioms representing the number of inferred resources required by the user task is 41. Typically, the inferred results might include resources such as nuclear imaging, intensive care unit, surgery unit, and other services (i.e. Distance To Emergency Ward, Time To Emergency Ward, Patient EHR Access Service, Weather Forecast Service, Traffic Status Service). The inferred facts are retrieved and filtered by using the SPARQL query language. In the above scenario, the SPARQL queries are set up to retrieve the healthcare institutions which have the material resources required by the task. The inferred solutions are then classified and filtered out according to the context of the user such as location or any other technical characteristics of the material resources needed by the patient transfer task.

4 Related Works

Resource Discovery is the process of finding the resources that meet and satisfy the user's requests. It includes resources description, organization, lookup, and selection. The resources discovery has been the subject of many research works which describe the possible approaches that can be employed for resources discovery. For example, a survey of resource discovery approaches in distributed computing environment has been presented by [1]. Additionally, an interesting comparative analysis of Resource Discovery approaches in grid computing has been realized by [2]. Also, a state of the art of semantic web service discovery has been provided by [3]. More specifically, a P2P-based infrastructure that leverages semantic technologies to support a scalable and accurate services discovery process has been proposed by [4]. The key idea presented in the latter work is the creation of an overlay network organized in several semantic groups of peers, each specialized in answering queries pertaining to specific application domains. Another approach, also based on P2P, has been proposed by [5] where the proposed architecture is organized as a hierarchical super-peer structure. In our research work we focus on the issue of resources mobilization and allocation after the discovery process. Thus, all the above cited approaches can be easily integrated in the RMA we propose. However, the question of how to adapt the resources mobilization process to the continuous changes of the resources conditions has been neglected by the above cited works. We have deeply considered this question in the RMA we present in this paper. In fact, the problem of the dynamic discovering of distributed, constantly changing, heterogeneous resources has been early introduced by [6, 7]. Several research efforts have been done to address the issue of automated resources allocation and to support decision making, especially in healthcare context-aware computing [8]. A service-oriented approach has been proposed by [9] for human tasks execution based on ontologies and agent technology. But these approaches have poorly considered the changing in resources metadata which is represented by two main aspects: (i) the continuous changing of QoS parameters; (ii) the conditions of the access and use of the different

resources by the user-tasks. These issues have been treated by the proposed RMA using a web-services based policy to update the inferred resources before filtering and allocating them to the user-tasks.

5 Conclusion

In this paper we propose an architecture model of a Resources Mobilization Agent (RMA) to solve the problem of continuous changing of the conditions of distributed heterogeneous resources. RMA is mainly based on a generic knowledge meta-model representing the basic entities of collaborative environments and their interrelations. The proposed RMA applies rule-based reasoning for resources inferring and utilizes multiple object oriented components for resources data processing including resources updating, filtering, ranking, and allocation. The key feature of the proposed Resources Mobilization Agent RMA resides in its capacity to automatically mobilize and allocate accessible and available resources needed for various user-tasks, taking into account the dynamic context of these resources. Thus, it contributes to enhance the quality of the exchanged information in context dependent situations.

References

1. Murugan, B.S., Lopez, D.: A Survey of Resource Discovery Approaches in Distributed Computing Environment. *International Journal of Computer Applications* 22, 44–46 (2011)
2. Sharma, A., Bawa, S.: Comparative Analysis of Resource Discovery Approaches in Grid Computing. *Journal of Computers* 3, 60–64 (2008)
3. Ngan, L.D., Kanagasabai, R.: Semantic Web service discovery: state-of-the-art and research challenges. *Personal and Ubiquitous Computing*, 1–12 (2012)
4. Di Modica, G., Tomarchio, O., Vita, L.: Resource and service discovery in SOAs: A P2P oriented semantic approach. *Int. J. Appl. Math. Comput. Sci.* 21, 285–294 (2011)
5. Eftychiou, A., Vrusias, B.: A Knowledge-Driven Architecture for Efficient Resource Discovery in P2P Networks. In: 2010 2nd International Conference on Intelligent Networking and Collaborative Systems (INCOS), pp. 467–472 (2010)
6. Liu, J.: World Wide Wisdom Web (W4) and Autonomy Oriented Computing (AOC): What, When, and How? In: Pal, S.K., Bandyopadhyay, S., Biswas, S. (eds.) PReMI 2005. LNCS, vol. 3776, pp. 157–159. Springer, Heidelberg (2005)
7. Haeupler, B., Pandurangan, G., Peleg, D., Rajaraman, R., Sun, Z.: Discovery through gossip. In: Proceedings of the 24th ACM Symposium on Parallelism in Algorithms and Architectures, pp. 140–149. ACM Press (2012)
8. Anya, O., Tawfik, H., Amin, S., Nagar, A., Shaalan, K.: Context-aware knowledge modelling for decision support in e-health. In: The 2010 International Joint Conference on Neural Networks (IJCNN), pp. 1–7 (2010)
9. Sasa, A., Juric, M.B., Krisper, M.: Service-Oriented Framework for Human Task Support and Automation. *IEEE Transactions on Industrial Informatics* 4, 292–302 (2008)

A New Proof System to Verify GDT Agents

Bruno Mermel and Gaele Simon

Abstract. The GDT4MAS model is dedicated to the formal specification of agents and multiagents systems. It has been presented in previous articles [7, 6]. The proof mechanism relies on *proof schemas* that generate *proof obligations*, that is to say first-order formulae that can be proven (in most cases) by an automatic prover (such as PVS). In this article, we present a new version of proof schemas that increase the number of proofs that can be performed.

1 Introduction

The GDT model has been first presented five years ago [7]. This model consists in a language to formally specify agents, a formal semantics, and a set of proof schemas to guarantee the correctness of the behaviour specified. It has been extended a few years later to specify and verify multiagent systems [6].

However, when applying the model and the proof system to concrete case studies, it appeared that true properties were unverifiable because necessary hypotheses were lacking. Thus, we propose here a new proof system that provides richer hypotheses.

In the next section, we briefly recall the main concepts of the GDT4MAS model. In section 3, we present the old proof schema principles, and we illustrate its weaknesses. Section 4 is dedicated to the presentation of the new proof schemas. Finally, section 6 concludes on this new proof mechanism.

2 The GDT4MAS Model

2.1 Main Concepts

In the GDT4MAS model, the MAS is described by an environment, mainly described by variables, and a population of agents evolving in this environment. Each agent is described as an instance of an agent type. As a consequence, in the rest of this section, after a short description of the notations we used, we begin by

Bruno Mermel · Gaele Simon

GREYC-CNRS - Universit du Havre, Le Havre, France

e-mail: {Bruno.Mermel,gaele.simon}@univ-lehavre.fr

describing the notion of agent type, and of agent behaviour. We invite the reader to refer to previous articles [7, 6] for more details on this model.

2.2 Properties Proven by the Method

The GDT4MAS method allows to prove several kinds of properties. We first prove invariant and liveness properties, at the agent-type level and at the system-level. We recall here that invariant properties are properties that must be always true, and that liveness properties are properties that must eventually be true. Moreover, the proof-system of the method verifies that goal decompositions are valid. In this article, we focus on the proof of decompositions and of invariant properties. This is the topic of the next section.

3 Previous Proof System

3.1 Principles

The main principles of the previous proof system consists in the following steps:

- A *context* is inferred for each node, in a top-down manner, and from left to right when oriented operators are used (like SeqAnd and SeqOr);
- A *gpf*, Guaranteed property in case of failure, is inferred for each NNS goal, in a bottom-up manner;
- A *proof schema* is used for each decomposition operator, in order to generate a *proof obligation*, that is to say a first-order formula, whose proof can be attempted by any first-order logic theorem prover.

3.2 Limits

3.2.1 Projections

The main limit of the approach defined above is that we use true projections, that cannot be computed automatically. A simplified version can be computed, by removing each term of the conjunctive normal form containing another variable than those concerning the projection. For instance, if a projection of the formula $\mathcal{F} \equiv (x = 2 \wedge y = 4 \wedge x = z + 2 \wedge z > 3)$ on x is required, we obtain $\mathcal{F}_x \equiv (x = 2)$. However, this formula is weaker than what we would like to obtain (In particular, the fact that x is greater than 5 is lost).

3.2.2 Weak Contexts

The context inferred for a goal is quite weak: it only depends on the execution of the previous goal: its satisfaction condition if the operator is a Seqand, or its guaranteed property in case of failure if the operator is a SeqOr. But more information

could be preserved from the context in which this previous goal is executed. For instance, if node n is decomposed into $n_1 SeqAnd n_2$, if the context of node n_1 is $x = y$, if x and y are internal variables and if the satisfaction condition of node n_1 is $sc_{n_1} \equiv x' = x + 1 \wedge y' = y + 1$, we know that $x = y$ when node n_2 is considered. But this cannot be automatically inferred by the context propagation rules.

3.2.3 Weak Proof Schemas

In proof schemas, some important hypotheses miss.

3.2.4 General Observations

Most of the weaknesses highlighted above can be bypassed by re-inforcing satisfaction conditions of goals. However, this is not a satisfying solution, because it requires more work to the developer, and it makes the specification more complicated. Thus, it appeared necessary to re-inforce the proof mechanism to solve these drawbacks.

4 The New Proof System

To define the new proof system, we need new notations, that are introduced in the following section.

4.1 Predicate Transformers

Notation 4.1 (At). Let f a predicate. $f[i]$ is a predicate where each non-subscripted variable in f is subscripted by i .

Example: $(x = y_0)[1] \equiv (x_1 = y_0)$.

Notation 4.2 (Between). Let f a predicate. $f^{i \rightarrow j}$ is a predicate derived from f where each unprimed and unsubscripted variable is subscripted by i and each primed variable becomes unprimed and subscripted by j .

Example: $(y' < x \wedge x' = x_0)^{1 \rightarrow 2} \equiv (y_2 < x_1 \wedge x_2 = x_0)$.

Notation 4.3 (Temporal switch). Let f a predicate. $f^{\rightarrow i}$ is predicate derived from f where each subscript is increased by i .

Example: $(x = x_1 \wedge y_2 = x_1)^{\rightarrow -2} \equiv (x = x_{-1} \wedge y_0 = x_{-1})$.

Notation 4.4 (Priming). Let f a predicate. If f contains at least one primed variable, then $pr(f) = f$. Otherwise, $pr(f)$ is the predicate derived from f where each unsubscripted variable is primed.

Examples: $pr((x = x_0)) \equiv (x' = x_0)$ and $pr((x = x')) \equiv (x = x')$.

Notation 4.5 (Stability). Let t and agent type with two internal variables via, vib and one surface variable vs (internal and surface variables are described in the next section). Then, when one agent a of this type is considered $stab^{i \rightarrow j}$ is the predicate $via_i = via_j \wedge vib_i = vib_j \wedge vs_i = vs_j$.

Notation 4.6 (Untemporalization). Let f a predicate. f^* is the formula f in which all subscripts of value x are removed.

Example: $(x_1 = x_2)^* \equiv x = x_2$.

Notation 4.7 (Invariant). Let A an agent situated in an environment \mathcal{E} . We write:

- i_A the invariant associated to the internal variables of the agent;
- $i_{\mathcal{E}}$ the invariant associated to the environment variables;
- $i_{\mathcal{E}A}$ the conjunction of i_A and $i_{\mathcal{E}}$.

4.2 Context Inference

In a context formula, it may be necessary to refer to the value of a variable in a previous state. In that case, the variable is subscripted by a negative integer. The value in the current state is represented by the variable name neither subscripted nor primed. So, the new context inference rules are the following:

$$\begin{cases} C_{N_1} = C_N \\ C_{N_2} = \left(\left((C_{N_1} \wedge sc_{N_1})^{0 \rightarrow 1} \wedge stab^{1 \rightarrow 2} \wedge i_{\mathcal{E}A}[1] \wedge i_{\mathcal{E}A}[2] \right)^{\rightarrow -2} \right)^{\rightarrow 0} \end{cases} \quad (1)$$

Using such rules, contexts are guaranteed to use neither primed variables nor variables with positive subscripts. Indeed, if its true for C_N :

- it is trivially true for C_{N_1} ;
- it is true for C_{N_2} because:
 - $(C_{N_1} \wedge sc_{N_1})^{0 \rightarrow 1}$ contains variables with negative subscripts from C_{N_1} , and variables subscripted by 0 and 1;
 - $stab^{1 \rightarrow 2}$ contains variables subscripted by 1 and 2;
 - $i_{\mathcal{E}A}[1]$ contains variables subscripted by 1;
 - $i_{\mathcal{E}A}[2]$ contains variables subscripted by 2;
 - From the previous facts, the formula

$$\left((C_{N_1} \wedge sc_{N_1})^{0 \rightarrow 1} \wedge stab^{1 \rightarrow 2} \wedge i_{\mathcal{E}A}[1] \wedge i_{\mathcal{E}A}[2] \right)^{\rightarrow -2}$$

- contains variables with negative or null subscripts;
- and so, C_{N_2} contains variables with negative subscripts or unprimed variables without subscripts.

4.3 Proof Schema

In this new proof mechanism, the context of the right subgoal, in the case of the SeqAnd operator, contains the essential properties that are guaranteed to be true when the right subgoal is considered. So, as the semantics of the SeqAnd operator is that, when the right subgoal is executed and succeed, the parent goal must be achieved,

the proof schema associated to the SeqAnd operator only consists in verifying that when the right subgoal succeeds in its context, then the parent goal also succeeds. This leads to the following proof schema:

$$(C_{N_2}[0] \wedge sc_{N_2}^{0 \rightarrow 1} \wedge i_{\mathcal{E}A}[1]) \rightarrow sc_N^{-2 \rightarrow 1} \quad (2)$$

This proof schema can be informally explained as follows:

- We consider the case where the second subgoal, N_2 , is executed, which occurs when its context C_{N_2} is verified. We assign to the state the number 0;
- We only consider the case where this subgoal succeed, reaching another state numbered 1, hence the formula $sc_{N_2}^{0 \rightarrow 1}$;
- We also know that in this new state, the environment variable and the agent variable are true. So we have the hypothesis $i_{\mathcal{E}A}[1]$;
- Finally, if all these hypotheses are true, the satisfaction condition of the parent goal must be established between the state in which the parent goal execution has begun (numbered -2) and the state in which the execution of the second subgoal ends (numbered 1).

4.4 Guaranteed Property in Case of Failure

In the previous proof system, GPF were inferred in a bottom-up manner from leaf goals. This characteristics presented at least two drawbacks:

- The GDT must be completely specified to be proven. Indeed, leaf goals must be known to infer GPF of intermediate goals, and these GPFs are required to prove the correctness of the GDT, as they are used in some proof schemas;
- The GPF inferred may be very complicated and may hold many non-necessary hypotheses, that may reduce the success rate of automatic provers.

So, in the new proof system, it is asked to the developper to explicitely give GPF of each node, as he has to do for satisfaction condition. This is not a harder work than specifying satisfaction condition, and it makes the system quite more compositional. However, it must be checked that the given GPF are entailed by the behaviour specified. Thus, we must give GPF inference rules and we have to verify that the GPF given by the developper is entailed by the GPF inferred from the GDT.

For the SeqAnd operator, the parent goal may fail either when the first subgoal fails (in that case, its gpf is verified in the context of the execution of the first subgoal) or when the second subgoal fails (the gpf of this subgoal is then verified in the context of the execution of this seconde subgoal). So we have:

$$\begin{aligned} \text{inf}gpf_N = \vee & (C_{N_1} \wedge gpf_{N_1}) \\ & (C_{N_2}[0] \wedge gpf_{N_2}[0])^{\times 2} \end{aligned} \quad (3)$$

We can notice that in the formula corresponding to the inferred gpf of a goal, unprimed variables correspond to the values of the variables when the execution of the

goal begins, and primed variables correspond to the values of the variables when the execution of the goal ends.

We have now to prove that the specified gpf is correct. Thus, for each NNS goal, we must establish that when the decomposition fails (that it to say, the inferred gpf is true), either the parent goal is achieved or its gpf is true. Hence the following proof schema:

$$\text{inf}gpf_N \rightarrow (sc_N \vee gpf_N) \quad (4)$$

5 Related Works

Several works deal with the formal specification of multi-agent systems, but just a few consider the formal verification of their specification. Moreover, most of these systems use model-checking [1, 8, 5]. The most recent and promising work in this area is surely the Agent Infrastructure Layer (AIL) [3] and its connection with AJPF (Agent JPF), an adaptation of the Java Path Finder model-checker. AIL is a kind of intermediate language that can be used to give a formal semantics to most BDI agent languages (such as AgentSpeak or MetateM). JPF is however not a very efficient model-checker. Although there are best model-checkers such as SPIN [4], this technique is not well-suited to massive multi-agent systems, where the state-space is too huge, especially because of the great interleaving possibilities between agent actions, and as a consequence, only systems with few agents manipulating boolean concepts can be proven. On the other hand, the system we propose make possible the automatic verification of huge systems, manipulating complex data using an any existing first-order theorem prover (it has been tested with PVS [9] and krt [2]).

6 Conclusion

The previous proof system for GDT4MAS agents was of course valid, but it was difficult to perform proofs with intuitive goal specifications: the satisfaction conditions that the developer had to give had to be really proof-oriented, and using the model was not really feasible by a developer who did not know all the details of the proof process. With the new principles presented in this article, the specification task is more independent from the proof process. Of course, the developer must be well-heeled with the predicate logic, but it is not necessary for him to understand how the proof works. Using the method is thus easier. Moreover, the number of proofs that can be performed is increased.

References

1. Bordini, R., Fisher, M., Pardavila, C., Wooldridge, M.: Model-checking AgentSpeak. In: AAMAS 2003, Melbourne, Australia (2003)
2. Clear-Sy: B for free, <http://www.b4free.com>
3. Dennis, L., Fisher, M., Webster, M., Bordini, R.: Model Checking Agent Programming Languages. Automated Software Engineering Journal 19(1), 3–63 (2012)

4. Holzmann, G.J.: The Model Checker SPIN. *IEEE Trans. Softw. Eng.* 23, 279–295 (1997), <http://dl.acm.org/citation.cfm?id=260897.260902>, doi:10.1109/32.588521
5. Kacprzak, M., Lomuscio, A., Penczek, W.: Verification of multiagent systems via unbounded model checking. In: Autonomous Agents and Multi-Agent Systems (AAMAS 2004) (2004)
6. Mermet, B., Simon, G.: GDT4MAS: an extension of the GDT model to specify and to verify MultiAgent Systems. In: Sichman, D., et al. (eds.) Proc. of AAMAS 2009, pp. 505–512 (2009)
7. Mermet, B., Simon, G., Zanuttini, B., Saval, A.: Specifying, verifying and implementing a MAS: A case study. In: Dastani, M., El Fallah Seghrouchni, A., Ricci, A., Winikoff, M. (eds.) ProMAS 2007. LNCS (LNAI), vol. 4908, pp. 172–189. Springer, Heidelberg (2008)
8. Raimondi, F., Lomuscio, A.: Verification of multiagent systems via ordered binary decision diagrams: an algorithm and its implementation. In: Autonomous Agents and Multi-Agent Systems, AAMAS 2004 (2004)
9. SRI International: PVS, <http://pvs.csl.sri.com>

Using Trusted Communities to Improve the Speedup of Agents in a Desktop Grid System

Lukas Klejnowski, Sebastian Niemann,
Yvonne Bernard, and Christian Müller-Schloer

Abstract. In open, technical, multi-agent based systems, self-interested agents can show behaviours that degrade the performance of the system. This can be countered by providing cooperation incentives. In this paper, we present a formalisation of delegation incentives for an open, agent-based, Desktop Grid system, based on decision trees. We then discuss reputation-based delegation strategies, as well as replication-based delegations strategies and focus on the incentives these strategies provide for agents to cooperate. We further show why we see room for improvement, and how this can be achieved with organisation-based delegation. We propose a delegation strategy based on *Trusted Communities* and present evaluation results for the comparison of these strategies with respect to the achieved average speedup in the system.

1 Introduction

In technical, multi-agent-based scenarios, the utility functions of the agents can be used to model the performance of the system, as well as the performance of the single involved participants. However, when these systems are open, we need to consider self-interested agents that do not care for the performance of other agents or the system as such. These agents have to be treated as blackboxes regarding their behaviour, as we cannot demand cooperative types of behaviour without violating agent autonomy. Instead, incentive mechanisms are used to influence the agent behaviour in a way that increases their willingness to cooperate.

In this paper, we discuss incentive mechanisms for such an open, multi-agent-based, technical system and argue how incentive mechanisms based on (a) trust and reputation, and (b) trust-based MAS-organisations, can be used to improve the performance of this system.

Lukas Klejnowski · Sebastian Niemann · Yvonne Bernard · Christian Müller-Schloer
Leibniz Universität Hannover, ISE-SRA, Appelstr. 4, Hannover, Germany
e-mail: {klejnowski,niemann,bernard,cms}@sra.uni-hannover.de

2 System Model

We study an open, distributed and volunteer-based Desktop Grid System (DGS) to which we refer as *Trusted Computing Grid (TCG)*, in the tradition of systems like XtremWeb [14]. The system is designed without central control and the applications regarded produce bag-of-task jobs, i.e. jobs being composed of independently processable work units (WUs). Each user can submit jobs to the system and is expected to volunteer its machine as worker for other users' work units.

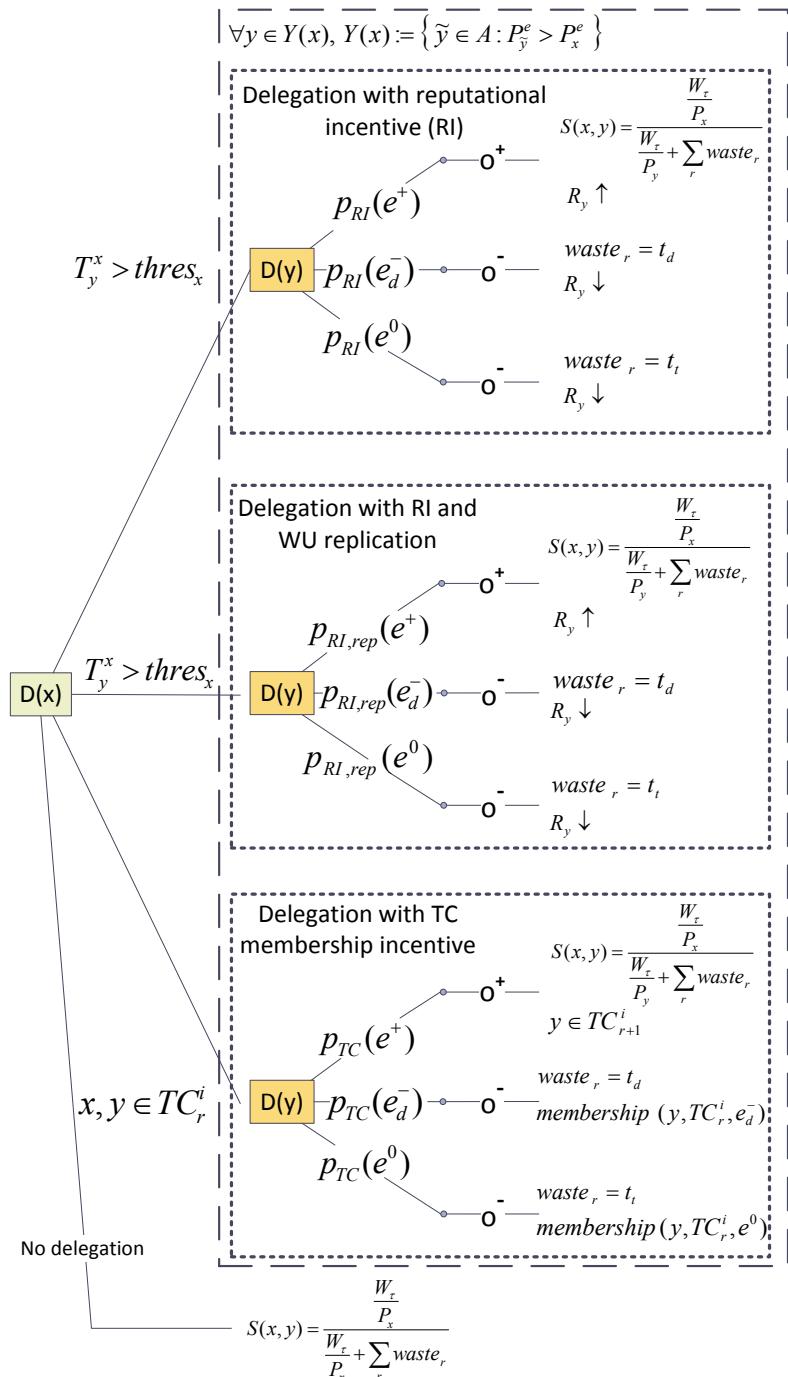
We use agents that are in charge of the grid client on the machines and make decisions on behalf of their users. The utility of the agents is based on their goal, to schedule single work units on available worker agents, such that they minimise the time it takes to receive valid results. This is formalised using metrics like *speedup*, *processing time* and *waste* (cf. e.g. [11] and [4]). Due to the open nature of the system, we have to deal with agents that show various types of behaviour (from altruistic to untrustworthy) in order to achieve their self-interested goal of scheduling their own jobs as efficiently as possible. According to the taxonomy of [10] and taking the resource perspective, we therefore classify the potentially participating agents of this Desktop Grid System as: egoistic, volatile, distributed over the internet, dynamic, faulty and heterogeneous. We therefore apply a Trust- and reputation system and let the agents build up trust relationships based on the outcome of their interactions as submitters and workers.

2.1 Submitter Decision Tree

In the following we present the agents' decision tree as depicted in Fig. 1, when taking the submitter perspective and searching for suited worker agents. An agent x from the agent society A that has an unprocessed work unit τ , first builds up the set $Y(x)$, containing each agent y that qualifies as worker based on its estimated performance P_y^e . P_y^e is dependent on the machine performance, the host and resource availability, and the work load of agent y . In sum, it represents an estimate of y 's *competence* as a delegation partner for x . The set $Y(x)$ is formally composed as follows:

$$Y(x) := \left\{ \bar{y} \in A : P_{\bar{y}}^e > P_x^e \right\} \quad (1)$$

The submitter then makes a decision $D(x)$ which delegation strategy is suited best for delegating τ to a worker agent $y \in Y(x)$. This is based on an assessment of y 's *willingness* to cooperate, being the probability that y is willing to invest an effort e . In a DGS, the positive outcome o^+ of the delegation is a state, where a valid result for τ exists, which can only be reached if y processes τ completely (effort e^+). Each other effort level produces a negative outcome o^- , meaning no or no valid result for τ . Here we discern between straight rejection to cooperate (e^0) and the attempt to produce o^+ through processing τ to a certain degree d , denoted by e_d^- . Additionally, y can be a malicious agent, choosing e_d^- to harm x . Whenever the outcome o^- was reached, x enters a new round r and either delegates to the next best worker in $Y(x)$ or processes τ itself when no agent is left in $Y(x)$. This is done until the outcome

**Fig. 1** Decision tree for submitter agents

is o^+ , with the assumption that x always produces o^+ when processing an own τ , hence always terminating.

In the following, we argue how x can choose a delegation strategy and what implications that choice has on the utilities and costs for both x and y . In the *TCG*, the utility of an agent is defined by the average *speedup* it is able to achieve for each task τ it submits. The speedup is defined as follows:

$$S(x, y) = \frac{\text{processing time}_{x, \tau}}{\text{processing time}_{y, \tau} + \text{waste}} = \frac{\frac{W_\tau}{P_x}}{\frac{W_\tau}{P_y} + \sum_r \text{waste}_r} \quad (2)$$

In eq. 2 we incorporate the size W_τ of a work unit and the actual worker performance P_y , and contrast this with the a posteriori measure of the submitter performance P_x and the summed waste in all rounds r with the outcome o^- . The waste_r is the time that τ was in processing in round r without an obtainable result¹. In case of e_d^- the amount of waste depends on the time t_d spent with the effort e_d^- , while the waste generated by the processing rejection e^0 is the time t_l needed for the transition to the next round $r+1$ and try to delegate τ to the next worker. In general, we observe $t_d \gg t_l$. It is obvious, that agents need to interact with cooperating workers as much as possible to increase their speedup.

The option to *not delegate* τ and process it by itself is always possible for x , but is only used as last resort when no competent or willing workers were found. This is because the speedup is dependent on x 's own performance P_x and can therefore never be greater than 1 for this strategy.

As long as there are willing workers available, the first delegation strategy with *reputation incentive* is preferred over self processing: We use a trust- and reputation system to rate the workers according to their effort². Positive outcomes result in y 's reputation gain (denoted by $R_y \uparrow$) and negative outcomes in the loss of reputation $R_y \downarrow$, with the amount of loss being dependent on the costs for x . We then let each agent x define a trust threshold thres_x such that the subjective trustworthiness T_y^x of y has to be greater than that threshold in order to consider y as suited worker. But how does a high effort pay off for the workers? The answer is through reciprocity: A worker y will make the decision $D(y)$ to reject a cooperation with x , if $T_x^y \leq \text{thres}_y$, thus if itself would not consider x as a suited worker for its own work units³. In this way reputation acts as incentive to cooperate and produces reputation loops between worker and submitter performance.

The second delegation strategy with *WU replication* (cf. Fig. 1) also applies the reputation incentive, however, here submitters try to decrease the probability for a low speedup to the disadvantage of the system: Instead of waiting until the result

¹ This is true for applications that do not allow checkpointing.

² In order to do this, we need to discern the outcomes o^- and o^+ . This is only possible if we have applications that produce work units whose results can be validated with far less effort than generated.

³ Agent y will also reject if its work load is already too high in order to maintain a condition in which it could process own work units with low costs if necessary.

for τ produced by a worker y can be validated, x generates replicas τ_i of τ and tries to delegate these to other workers. As soon as a valid result is returned, it can cancel the processing of the remaining replicas. This is an appropriate strategy from the cost perspective of a submitter, as replication generates hardly any additional overhead. However the work load in the system is raised by the replication factor until o^+ is reached: This not only reduces the probabilities $p_{RI,rep}(e^+)$ and $p_{RI}(e^+)$, as $D(y)$ depends on y 's work load, for other submitting agents, but also for x itself, as work units τ come in bursts (jobs). In the long term the speedup of x can therefore even decrease. On the worker side, wasteful processing of replicas blocks the worker. This reduces the opportunities to work for agents that could reciprocate and hence counters the effects of the replication incentive. When applying this strategy, submitting agents should therefore take the work load of suited workers into account when making the decision $D(x)$.

2.2 Discussion

Delegation with reputation incentives introduces a problem addressed in [7]: With agents accumulating trustworthiness through cooperation, a situation can develop where there is too much trust (*over-confidence*) to quickly react to changing behaviour. This becomes clear when regarding $thres_x$: A highly trustworthy agent can have a strong negative impact on the speedup of other agents, if it starts to defect, because it will only be ruled out as worker if successive reputation losses lower its trustworthiness below the threshold. Additionally, too much trust in a worker y means that a submitter x subsequently invests more validation and monitoring costs to evaluate the worker than effectively necessary.

We therefore argue that we need an additional delegation strategy that counters these issues and hence allows for faster reaction to changing behaviours of trustworthy agents and lower costs for submitters. Additionally, the incentive provided by this new delegation strategy has to be stronger than the other incentives, in order to enable worker agents to participate in these interactions. In the following, we propose such a strategy, with delegation based on the membership in an organisation called *Trusted Community*, as introduced in [19].

2.3 Delegation within Trusted Communities

Trusted Communities (TCs) are formed and joined by self-interested agents with strong mutual trust relations and the purpose to increase their personal utility. TCs are maintained by management actions delegated to a designated member called *Trusted Community Manager (TCM)*, having the goal to preserve and optimise the composition and stability of this organisation. This organisation provides performance benefits for their members by improving interaction efficiency, information sharing and cooperation between the agents.

Agents become members of a TC depending on the strategy the TCM has with respect to the composition. In general, the realisation of this strategy is based on the

observed trustworthiness of agents in the society. When becoming a member of a TC TC_r^i , an agent y makes a contract with the TCM and all other members. This contract is based on the notion of *kinship* and states that, if chosen as delegate by an other member x in the round r , the agent y commits to provide the effort e^+ . In case of a contract violation due to effort e^0 or e_d^- , the TCM applies a *member control strategy* on the TC_r^i , such that it decides on the membership of y , formalised by:

$$TC_{r+1}^i = \text{membership}(y, TC_r^i, e^0) \in \{TC_r^i, TC_r^i \setminus \{n\}\}^4 \quad (3)$$

This contract provides strong incentives to invest the effort e^+ , as agents can rely on other members also providing them with the same effort. On the other hand, the validation of a contract violation through e^0 is cheap in terms of costs and fast as no processing time is involved. The validation of a violation via e_d^- is obviously more expensive, but can be fairly distributed among the members of a TC. In sum, we thus expect $p_{TC}(e^+) > p_{RI}(e^+)$. This means that the usage of a TC-based delegation strategy is more suited to increase the speedup of agents, than the usage of a reputation-based delegation strategy, as will be shown in section 3. Additionally, this incentive mechanism avoids the problems of too much trust and over-confidence, as we do not need to rate workers with trust- and reputation values, but can react to uncooperative behaviour immediately (with membership loss).

Besides controlling members in the worker role, the *TCM* can control members in the submitter role. This is realised by a monitoring scheme that is transparent to the submitters and does induce only low costs on the members: Workers in the TC report information on the work units accepted for processing to the *TCM*, which then is able to detect whether submitters have applied WU replication. Again this can be sanctioned via the *member control strategy* and works as incentive to cooperate. This monitoring is of course not applicable in general in the whole system, but needs a scaled environment like the TC.

The applicability of the *Trusted Community* delegation strategy is dependent on the trust-relationships within the agent society: Only where mutual trust builds up through the reputation incentive delegation strategy, TCs can be formed. Besides, these relationships often develop between clusters of agents that can partition the society into groups of cooperating agents. We consider this by allowing the formation of several Trusted Communities. In the following, our evaluation of the benefit of the delegation strategies is presented.

3 Evaluation

We have conducted experimental evaluations in simulations of the Trusted Computing Grid, to show that the formation and operation of Trusted Communities as delegation control organisation can improve the speedup of the member agents and hence the average speedup of cooperative agents in the system. As experimental setup we have used an agent society with 100 agents with heterogeneous machine

⁴ Analogous for effort e_d^- .

performance and behaviour with respect to the preferred effort level (adaptive to system perception, free-riding or selfish). Each agent produced in average 21 grid jobs composed of an average of 16 WUs. Thus the minimal number of $D(s)$ -decision evaluations was 33600 during the experiment runtime. The experiments were conducted with the option to have either no TC formation, allow for a single TC to form, or no restriction to the number of formed TCs. In the case of no TC formation, agents have used the delegation strategy with the reputation incentive. We then have conducted 20 runs per option and measured the variance in the achieved speedup as depicted in Fig. 2.

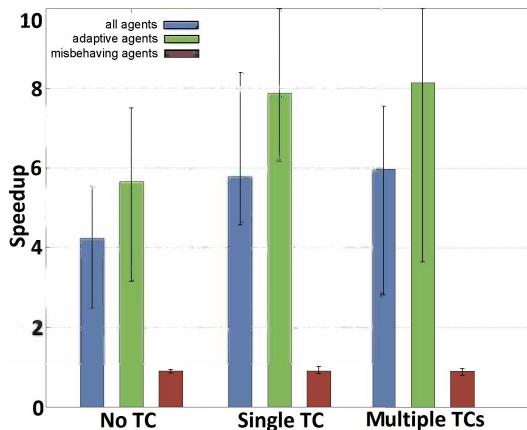


Fig. 2 Speedup gain through the application of Trusted Communities

The results show that, due to the TC membership incentive based on the kinship contract, TCs are able to increase the speedup of member agents. Additionally, the formation of several TCs is suited to further improve the results which shows the scalability of the approach. The variation of the single run results is relatively high, indicating that the composition and performance of the TCs very much depends on the system state in the *TCG*. However, we observe also a high variation for the system without TC formation, such that we attribute this to the varying agent behaviour dynamics. What the incentives have in common is, that notoriously uncooperative agents are isolated, such that they are forced to apply the *no delegation* strategy, resulting in a low speedup.

4 Related Work

We base our methodology on the generic decision tree for trustor agents presented in [6] and extend this work by applying it to the Desktop Grid domain, as well as propose an additional delegation strategy based on our work on the multi-agent

organisation *Trusted Community* (cf. [19]). Similar approaches to MAS organisations can be found in [16], [5] and [20]. In these approaches, we find that some requirements for the open Desktop Grid System we evaluate here are not met. In particular, *congregations* by [5] do not incorporate any notion of trust. In open distributed computing systems, trust and reliability play an essential role in the partitioning of participants into reliable and unreliable hosts (cf. e.g. [13], [3], [21]). *Clans*, as presented in [16], are best described as *congregations with trust* (cf. [18]). As such, clans provide the previously mentioned support for trust-based interaction modelling. However, clans are a purely decentralised approach without hierarchy. We assess this to be detrimental in systems environments where agent behaviour is highly dynamic, as we assert single agents the potential to damage the stability of an MAS-organisation if no coordinated self-management is performed. We therefore incorporated hierarchy in the TC design. Additionally, a strict reliance on a trust management system can also be detrimental if over-confidence builds up. In this we follow the argumentation in [7] regarding negative consequences of over-confidence. The way we implement TC membership as favourable compared to non-association is based on the ideas of incentive compatible design (cf. e.g. [22]). We especially follow the argumentation that agents are rational with respect to maximising their utility function (cf. e.g. [8]) and refer to a concrete utility definition (*speedup*, cf. e.g. [11],[23]) for an open Desktop Grid. The incentives for Desktop Grid agents provided by TC membership are a reciprocity-based approach, comparable to approaches discussed in e.g. [15], however transferred to agents representing the users. Again, the conclusion by the authors that these types of approaches are vulnerable to collusive behaviour, confirms our approach of hierarchical management of a TC as introduced in [19] and scheduled for future work. As for our application scenario, the main classification according to the system perspective in the taxonomy presented in [10], is that of *volunteer-based, distributed, P2P-based, internet-based Desktop Grid System*. Additionally, taking the resource perspective, we can conclude that participants are *egoistic, volatile, distributed over the internet, dynamic, faulty and heterogeneous*, comparable to e.g. the approach presented in [9]. This constitutes a competitive system with great uncertainty regarding expected performance and can be addressed by the application of trust as incentive-criterion, as for example in [12], [23], [17] and [13].

Additional coverage of MAS organisations in similar scenarios has also been considered in [24], [1] and [25].

5 Conclusion and Outlook

We presented an approach to formalise the delegation decision of agents in an open Desktop Grid System and described how a delegation strategy with an reputation incentive can improve the average speedup of these agents. We then discussed the drawbacks of this approach and proposed a new approach based on the application of the MAS-organisation *Trusted Community*. We showed how this delegation strategy can counter the drawbacks of a pure reputation strategy and presented the

evaluation results. These show that the application of TCs leads to a higher average speedup in the system. Future work will focus on exploiting this incentive mechanism further by using trustworthiness prediction, as in [2], to allow for earlier TC formation and thus less over-confidence. Additionally, we will compare the results with the performance of a delegation within *clans*.

Acknowledgements. This research is funded by the research unit “OC-Trust” (FOR 1085) of the German research foundation (DFG).

References

1. Abdallah, S., Zhang, H., Lesser, V.: The role of an agent organization in a grid computing environment. In: Proceedings of the 14th Int Conference on Automated Planning and Scheduling, Workshop on Planning and Scheduling for Web and Grid Services (2004)
2. Anders, G., Siebert, F., Steghöfer, J.-P., Reif, W.: Trust-Based Scenarios - Predicting Future Agent Behavior in Open Self-Organizing Systems. In: Proceedings of the 7th Int. Workshop on Self-Organizing Systems, IWSOS 2013 (2013)
3. Andrade, N., Brasileiro, F., Cirne, W., Mowbray, M.: Discouraging Free Riding in a Peer-to-Peer CPU-Sharing Grid. In: Proceedings of the 13th IEEE Int. Symposium on High Performance Distributed Computing, pp. 129–137. IEEE Computer Society, Washington, DC (2004)
4. Anglano, C., Brevik, J., Canonico, M., Nurmi, D., Wolski, R.: Fault-aware scheduling for Bag-of-Tasks applications on Desktop Grids. In: 2006 7th IEEE/ACM Int. Conference on Grid Computing, pp. 56–63. IEEE (2006)
5. Brooks, C., Durfee, E.: Congregation formation in multiagent systems. Autonomous Agents and Multi-Agent Systems 7(1) (2003)
6. Burnett, C., Norman, T.J., Sycara, K.: Trust decision-making in multi-agent systems. In: Proceedings of the 22nd Int. Joint Conference on Artificial Intelligence, vol. 1, pp. 115–120 (2011)
7. Castelfranchi, C., Falcone, R.: Trust Theory - A socio-Cognitive and Computational Model. John Wiley & Sons Ltd. (2010)
8. Centeno, R., Billhardt, H.: Using incentive mechanisms for an adaptive regulation of open multi-agent systems. In: Proceedings of the 22nd Int. Joint Conference on Artificial Intelligence, Barcelona, Spain, vol. 1, pp. 139–145 (2011)
9. Chakravarti, A., Baumgartner, G., Lauria, M.: The organic grid: self-organizing computation on a peer-to-peer network. In: Proceedings of the Int. Conference on Autonomic Computing, pp. 96–103. IEEE (2004)
10. Choi, S., Buyya, R., Kim, H., Byun, E.: A Taxonomy of Desktop Grids and its Mapping to State of the Art Systems. Technical report, Grid Computing and Distributed Systems Laboratory, The University of Melbourne (2008)
11. Cremonesi, P., Turrin, R.: Performance models for desktop grids. In: Proceedings of the 10th Int. Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS). Citeseer (2007)
12. Domingues, P., Sousa, B., Moura Silva, L.: Sabotage-tolerance and trustmanagement in desktop grid computing. Fut. Gener. Comput. Syst. 23(7) (2007)
13. Dyson, J., Griffiths, N., Lim, H., Jarvis, S., Nudd, G.: Trusting agents for grid computing. In: 2004 IEEE Int. Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583) (2004)

14. Fedak, G., Germain, C., Neri, V., Cappello, F.: XtremWeb: A generic global computing system. In: IEEE/ACM Proceedings of the 1st Int. Symposium on Cluster Computing and the Grid. IEEE Computer Society (2001)
15. Feldman, M., Chuang, J.: Overcoming free-riding behavior in peer-to-peer systems. ACM SIGecom Exchanges 5(4), 41–50 (2005)
16. Griffiths, N.: Cooperative clans. *Kybernetes* 34(9/10) (2005)
17. Griffiths, N.: Task delegation using experience-based multi-dimensional trust. In: Proceedings of the 4th Int. Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2005, p. 489. ACM Press, New York (2005)
18. Horling, B., Lesser, V.: A Survey of Multi-Agent Organizational Paradigms. *The Knowledge Engineering Review* 19(4), 281–316 (2005)
19. Klejnowski, L., Bernard, Y., Anders, G., Müller-Schloer, C., Reif, W.: Trusted Community - A Trust-Based Multi-Agent Organisation for Open Systems. In: Proceedings of the 5th Int. Conference on Agents and Artificial Intelligence (ICAART), Barcelona, Spain (2013)
20. Mathieu, P., Routier, J.-C., Secq, Y.: Principles for dynamic multi-agent organizations. In: Kuwabara, K., Lee, J. (eds.) PRIMA 2002. LNCS (LNAI), vol. 2413, pp. 109–122. Springer, Heidelberg (2002)
21. 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)
22. Ramchurn, S.D., Huynh, D., Jennings, N.R.: Trust in multi-agent systems. *The Knowledge Engineering Review* 19(01), 1–25 (2004)
23. Shudo, K., Tanaka, Y., Sekiguchi, S.: P3: P2p-based middleware enabling transfer and aggregation of computational resources. In: Proceedings of the IEEE Int. Symposium on Cluster Computing and the Grid CCGrid 2005, vol. 1 (2005)
24. Thabet, I., Bouslimi, I., Hanachi, C., Ghédira, K.: A multi-agent organizational model for grid scheduling. In: O'Shea, J., Nguyen, N.T., Crockett, K., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2011. LNCS, vol. 6682, pp. 148–158. Springer, Heidelberg (2011)
25. Wang, Y., Vassileva, J.: Trust-based community formation in peer-to-peer file sharing networks. In: Proceedings of the 2004 IEEE/WIC/ACM Int. Conference on Web Intelligence, WI 2004. IEEE Computer Society, Washington, DC (2004)

High-Volume Data Streaming with Agents

Lars Braubach, Kai Jander, and Alexander Pokahr

Abstract. Agent technology is in principle well suited for realizing various kinds of distributed systems. Nonetheless, the specifics of agent technology render it difficult to implement data driven systems, even though these represent an important class of today's applications including e.g. audio or video streaming and file transfer. One central reason why agents are not especially helpful for these kinds of applications is the message-driven high-level interaction of agents. These kinds of interactions are meant to support high-level coordination of agents via speech act based actions, not transport of high volume binary data. Hence, currently agent developers need to resort to standard communication technologies like TCP or UDP to transport data, but this requires complex management of low-level aspects like connection handling at the application level and additionally might pose problems due to the requirement of opening additional ports. To better support also binary data transfer a new streaming approach is proposed, which introduces virtual connections between agents. Usage resembles established input and output streaming APIs and lets developers transfer data between agents in the same simple way as e.g. a file is written to hard disk. Furthermore, virtual connections allow for failure tolerant transmission on basis of multiplexed data. The usefulness of the approach will be further explained with a real-word example application from the area of business intelligence workflows.

1 Introduction

Although multi-agent systems provide concepts for realizing various kinds of distributed systems, applications with a data centered background are not well supported due to the focus on high-level messaging among agents. In order to build applications that need to transfer huge amounts of binary data between agents two different approaches are available. First, one can directly employ communication

Lars Braubach · Kai Jander · Alexander Pokahr
Distributed Systems and Information Systems Group,
University of Hamburg, Hamburg, Germany
e-mail: {braubach, jander, pokahr}@informatik.uni-hamburg.de

libraries e.g. TCP streams. This has the disadvantages of being forced to handle lower-level and in many cases intricate communication aspects at the application level. Second, one can rely on the existing message based communication mechanisms of agents and use them to transfer binary data. The primary problem of this approach is that will not work with arbitrary large data as it cannot be held in main memory completely and additionally performance degradation is likely to occur.

In order to motivate the requirements of a streaming approach we have analyzed an ongoing commercial project called DiMaProFi (Distributed Management of Processes and Files) in the area of business intelligence that is conducted together with the company Unique AG.¹ The main objective of the project is to build a distributed workflow management system for ETL (extraction, transformation, loading) processes. These processes are in charge of moving and transforming huge amounts of data from different locations to a data warehouse, in which they are loaded to allow further inspections by domain experts. The workflow tool must allow specifying very different kinds of workflows as these are completely dependent on the concrete customer. From this scenario the following requirements were deduced:

- *Location transparent addressing*: Addressing should be done between agents and should be location transparent, i.e. it should be possible to transmit data between agents without knowing their location. In the project, the ETL processes are realized as agents. Here, it is often necessary to copy files to the executing workflow, e.g. if a subprocess is executed by a different node, data has to be transferred to it.
- *Infrastructure traversal*: Data transfer must be able to cope with the existing infrastructure characteristics and restrictions. This means that e.g. firewall settings might constrain the ability to open new connections for transmissions. As a result, existing communication channels have to be reused and shared. The infrastructures on which DiMaProFi is deployed depends on the customer, ranging from banking scenarios with strong restrictions to internet providers with fewer security policies but distributed networks. Constraints can therefore vary to a high degree.
- *Failsafe connections and heterogeneous multihoming*: Data transfer between agents should be as failsafe as possible and use all available means to reach the other agent, for example during connection breakdowns etc. Furthermore, DiMaProFi deals with big data. This means that it is crucial to avoid complete retransmissions of large files if parts already have been successfully transferred.
- *Non-functional properties*: The quality of service characteristics, i.e. non-functional properties, of the transfer should be configurable. Important properties include e.g. security, reliability, and priorities. In the workflow scenario, customers often execute different kinds of ETL processes at the same time. As these processes have different deadlines, it is important to allocate execution resources according to these deadlines. Part of these resources are non-functional properties of data transfers.

¹ <http://www.unique.de/>

In this paper an approach will be presented that addresses these requirements with a distributed streaming concept based on virtual connections. The remainder of this paper is structured as follows. The next Section 2 presents the approach itself and its implementation. Thereafter, Section 3 illustrates the usage of the approach with the ETL workflow application. Section 4 compares the proposed approach to existing solutions and Section 5 gives some concluding remarks and a short outlook to planned future work.

2 Data Streaming Approach

The requirements of the last section have been carefully analyzed and strongly influenced the design of the streaming architecture presented later. Here the findings are shortly summarized according to the already introduced categories. As an additional point the agent integration has been added as the characteristics of agents also determine the set of possible solutions.

- *Location transparent addressing:* This implies that a connection should have an agent as start and endpoint. Furthermore, the streaming mechanism should be enabled to use the existing agent platform addressing to locate the target agent platform.
- *Infrastructure traversal:* In order to cope with different environments and security settings, the solution uses existing communication channels for multiple streams, i.e. multiplex the data.
- *Failsafe connections and heterogeneous multihoming:* Failsafe connections require that streams should be able to communicate via different underlying transport connections, i.e. the mechanism must be able to dynamically switch in case of a breakdown. Moreover, the required intelligent usage of underlying transports requires a layered approach in which an upper coordination layer selecting and managing the underlying transports.
- *Non-functional properties:* The coordination layer has to consider the properties when selecting among different transport options (e.g. whether a transport is encrypted, authenticated etc.)
- *Agent integration:* The streaming mechanism should be accessible to the agents in a non-disruptive way, i.e. streams being an option in addition to the traditional message sending approach.

2.1 General Architecture

In Fig. 1 the streaming architecture is depicted. From a high-level view an agent should be enabled to directly use input and output connection interfaces - in addition to sending messages - to directly stream binary data to / or receive data from another agent. The figure also shows that the basic envisioned architecture relies on the standardized FIPA platform architecture [6] in the sense that it is assumed that on each agent platform a MTP (message transport service) exists that is capable of

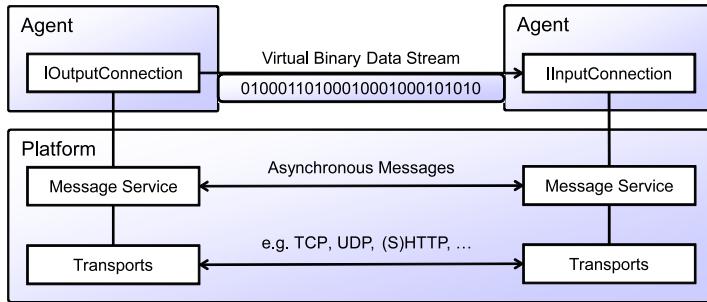


Fig. 1 Stream architecture

sending asynchronous messages to agents of the same and other platforms. For this purpose it makes use of different transports, which utilize existing communication technologies such as TCP, UDP or HTTP to transfer the data.

2.2 Stream Usage

In order to better understand the envisioned usage from an agent perspective, in Fig. 2 the important streaming interfaces are shown. Each connection (IConnection) has a connection id as well as two endpoints, an initiator (agent) as well as a participant (agent). Each side is free to close the stream unilaterally at any point in time. The other side will be notified of the termination via a corresponding exception. Furthermore, each connection may be initialized with non-functional properties consisting of key value pairs.

An output connection (IOURLConnection) is used to write binary data in chunks to the stream (write). As it is often the case the sender and receiver cannot process the stream data at the same speed a new mechanism has been introduced to inform the output side when the input side is ready to process more data (waitForReady).

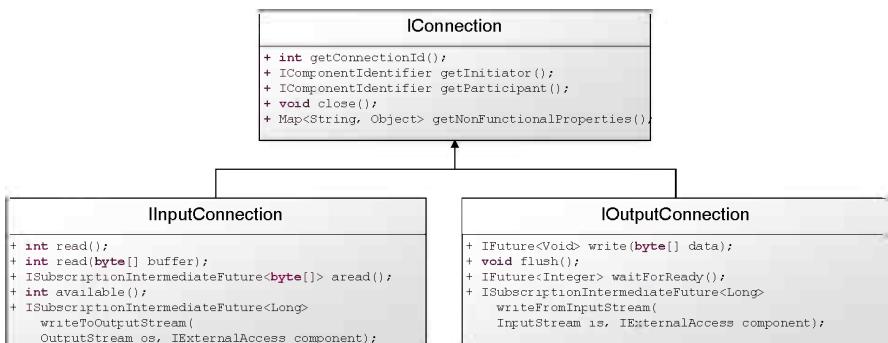


Fig. 2 Stream Interfaces

Finally, also a convenience method has been introduced that allows for automatically processing a Java input stream by reading data from it and writing it into the output connection until no more data is available (`writeFromInputStream`).

The input connection (`IInputConnection`) offers methods to read data from the stream. These methods include variants for reading a single byte, as well as a complete buffer of bytes. Before calling these methods it can be checked how much data is currently available at the stream (`available`). Moreover, it is possible to register a callback at the stream and automatically get notified when new data is available (`aread`). The input connection also possesses a method for connecting to standard Java streams. In this respect, the input connection allows for automatically writing all incoming data to a Java output stream (`writeToOutputStream`).²

2.3 Low-Level API

Besides the functionality an agent uses to send and receive data from the stream the question arises how streams are created by the initiator and received by the participant of a connection. For the first part the interface of the message service has been extended with two methods that allow for creating virtual connections to other agents. The method signatures are shown in Fig. 3. The caller is required to provide the component (i.e. agent) identifier of the initiator and the participant of the connection. Furthermore, optionally additional non-functional properties can be specified which have to be safeguarded by the message service during the stream's lifetime. As result of the call the corresponding connection instance is returned.

```

01: IFuture<IOutputConnection> createOutputConnection(IComponentIdentifier initiator,
02:           IComponentIdentifier participant, Map<String, Object> nonfunc);
03:
04: IFuture<IInputConnection> createInputConnection(IComponentIdentifier initiator,
05:           IComponentIdentifier participant, Map<String, Object> nonfunc);

```

Fig. 3 Extended message service interface

An agent that is used as participant in one of the create connection methods is notified about the new connection via the hosting platform. This is done via a new agent callback method (`streamArrived`) that is automatically invoked whenever a new stream is created. Behind the scenes the platform of the initiator contacts the platform of the participant and creates the other end of the connection at the target side. This connection is afterwards passed as parameter to the `streamArrived` method

² Please note that in contrast to Java streams all connection interfaces are non-blocking although the method signatures look similar. Blocking APIs are not well suited to work with agents as these are expected to execute in small steps to remain reactive. An agent that would directly use a blocking stream method could not respond to other incoming requests during it waits for the blocked call to return. In the interfaces different future types are used to render them asynchronous. A future represents a placeholder object that is synchronously returned to the caller. The real result will be made available to the caller once the callee has finished processing and set the result in the future [7].

call. Having received such a callback the receiving agent is free to use as it deems appropriate. Of course, it can also do nothing and ignore such incoming stream connection attempts.

2.4 High-Level API

For active components [5], which in brief are extended agents that can expose object oriented service interfaces, another more high-level API has additionally been conceived. As interactions with active components are primarily based on object-oriented service calls, it becomes desirable to be able to use streams also as parameters in these service calls. Using the high-level API an active component can declare streams as arbitrary input parameter or as the return value of a call. This allows for passing a stream directly to another agent solely by calling a service method.

Realization is complicated by the fact that method signatures contain the expected connection type of the callee but not of the caller. This means that a caller that wants to stream data to the callee has to create an output connection and write data to it but has to pass an input connection as parameter to the service call for the callee to be able to pull the data out of the stream. To solve this issue new service connection types have been introduced, which allow for fetching the corresponding opposite connection endpoint.

Support of non-functional properties has also been mapped to the high-level API. As these aspects should not be part of method signatures, that are meant to be functional descriptions, an annotation based approach has been chosen. For each supported non-functional property a corresponding Java annotation exists that can be added to the method signature of a service, i.e. `@SecureTransmission` can be used to ensure an encrypted data transmission.

2.5 Implementation Aspects

The proposed concept has been implemented as part of the Jadex platform [5]. The implementation distinguishes different responsibilities via different layers (cf. Fig. 1). On the top layer, the input and output connections ensure that streams comply with the functional and non-functional stream requirements. These requirements are addressed by a virtual stream control protocol, which is based on well-established TCP concepts.³

An output connection sends stream data in form of packets with a fixed size via the underlying message service. Thus packets, provided by the application layer, are created by either joining too small data chunks or by fragmenting larger ones depending on their size. The output connection keeps track of lost packages and resends them if necessary. Furthermore, the connection realizes flow control by using a sliding window that adapts the sender's connection speed to the speed of the

³ A virtual connection has to provide the requested service guarantees regardless of the existing infrastructure and underlying communication stack. For this reason it is necessary to reconstruct many aspects of TCP and other protocols on the upper layer.

receiver. Connection set-up and teardown is handled via specific handshake messages. The input connection receives and collects packets to forward them to the application level in the correct order.

The underlying message service has been extended to manage virtual connections and support sending messages belonging to the virtual connection protocol. Whenever the API is used to create a virtual connection (cf. Fig. 3) the message service internally creates a connection state at both connection sides and also starts a lease time based liveness check mechanism to ensure that the other connection side is still reachable. In case the lease times indicate that the connection has been lost it is closed unilaterally. The transport layer itself does not need changes have to support streaming.

3 Case Study

In this section the streaming approach is further explained by dint of the already introduced DiMaProFi workflow management project with Uniique AG. Customer-specific ETL processes are generally based on files which need to be loaded, transformed and then written into the customer's data warehouse. As an example a simplified version of a real world ETL banking process is used in the following. Here, source files are deposited in a special folder monitored by a process on a file server. Since the file sizes are considerable and the ETL process requires a substantial amount of processing time, the transformation processes are executed on different machines in the network in parallel for increased performance. The file server and the data warehouse are separated by a firewall which allows only certain traffic to pass.

Fig. 4 shows an example for such a process. When a customer file is stored on the file server, the monitoring process is notified and initiates the ETL process on a remote machine. The process requests the binary stream for the file server (fetch customer file) and stores the file in a temporary folder on the target machine. Then the received data is cleaned up with respect to the contained address data and thereafter two parallel transformations are performed on the same output data. The resulting data sets are written in parallel into the data warehouse. This process is performed in parallel on multiple machines for each file that has been deposited on the file server.

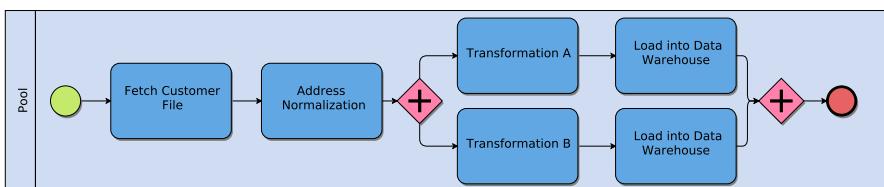


Fig. 4 An ETL process loading a file, transforming and writing it to the data warehouse

```
// File service service method
01: public IFuture<IInputConnection> fetchFile(String filename);

// Fetch file task except
01: IInputConnection icon = fileservice.fetchFile().get();
02: FileOutputStream fos = new FileOutputStream(tmpfolder+"/"+filename);
03: icon.writeToOutputStream(fos, agent).get();
04: fos.close();
```

Fig. 5 Code excerpts of fetching a remote file

The code of the fetch customer file task, which uses the high-level streaming API, is depicted in Fig. 5. It consists of the (reduced) interface of the file service, which offers a method `fetchFile()` to retrieve a remote file.⁴ As parameter the method takes the local file name and as result it delivers an input connection that can be used to download the file data. The code for downloading the file is shown below. First the input connection is obtained by calling the `fetchFile()` service method of the file server (line 1). Afterwards a file output stream for the temporary file is created (line 2) and the whole file content is automatically written to this file output stream by calling `writeToOutputStream()` (line 3). Please note, that this method takes the agent as argument as it executes the stream reading as agent behavior. The `get()` operation blocks until no more data is received all data has been written to the file. Finally, the stream is closed.

4 Related Work

As mentioned in Section 2, powerful streaming support includes a number of requirements that are not generally part of agent communication systems and network communication is often used to supplant it. However, overlay networks may offer an approach unrelated to agent that promises to meet some of the requirements. As a result, three basic categories are considered: agent communication, direct network communication and use of overlay networks, examples for each are shown in Fig. 6.

	Agent Message Communication Jade/JMS	Network Communication TCP Connection	Overlay Networks	
			SpoVNet	RON
Streaming Support	-	+	-	+
Agent Integration	+	-	-	-
Location-transparent Addressing	+	-	+	-
Infrastructure Traversal	-	-	+	+
Heterogeneous Multi-Homing	+	-	+	-
Failsafe Connections	-	-	o	o
Automatic Configuration	-	-	-	-
Non-functional Properties	-	-	+	-

Fig. 6 Streaming support requirements and support by different approaches

⁴ The `get()` method is part of the future API and causes the call to block until the asynchronous invocation has returned.

Streaming has not been a priority for agent systems. The traditional approach for agent communication centered around the exchange of speech act based messages, e.g. in JADE [2], which typically uses HTTP to transfer messages. Messages free the agents of low-level communication details and provide a form of location-transparent addressing. This approach is suitable for the exchange of small amounts of data, however, the lack of explicit streaming support forces agents to send bulk data in large messages, which can unnecessarily block the agent, or the messaging layer of the agent platform.

It is thus often suggested to use direct network connections such as TCP sockets for streaming and bulk transfer [3]. However, this forgoes the advantage of location-transparent addressing and burdens the agent with a number of low-level tasks, among them networking concerns such as firewall traversal. Furthermore, calls to such communication channels are often blocking, forcing intra-agent multithreading and increasing risks of data loss and deadlocks. In addition, if the connection is interrupted, recovery is difficult and if the chosen protocol like TCP is unavailable, the agent is unable to stream data at all. Both network connection and agent messaging only provide little support for non-functional features. While network connections often have QoS implementations, their configuration is hard and must be done at the system level. Application-level QoS-features such as the IPv4 type-of-service (TOS) field are generally ignored by routers.

An alternative consists in using overlay networks, which often bundle some of the required features such as (heterogeneous) multi-homing, location-transparent addressing and infrastructure traversal. While overlay networks do not provide specific support for agents, they often include a number of useful features. For example, Resilient Overlay Networks (RON) [1] allows streaming by tunneling TCP connections and allows multi-homing and, given an appropriate configuration, infrastructure traversal by relaying communications using other nodes. However, the multi-homing is not heterogeneous and thus connections are only failsafe in a limited sense. Furthermore, the addressing issue is not resolved and non-functional properties are unsupported. The overlay network framework Spontaneous Virtual Networks (SpoVNet) [4] does support both: location-transparent addressing through unique identifiers and specification of non-functional properties. It also provides some means for heterogeneous multi-homing using multiple means provided by underlays to transfer messages. However, it does not provide streaming support.

In general, overlay networks show the most promise toward providing a solution for the requirements but cover only a subset of the required feature set. Combining multiple such networks may be possible; however, this is hampered by problems such as integration of different programming languages.

5 Conclusions and Outlook

In this paper a concept and implementation has been presented that allows agents to stream binary data without consideration of detailed communication aspects. For this purpose two different APIs have been described. The low-level API enables creation of virtual streams to other agents via the message service and the

high-level API permits stream utilization as normal service parameters and return values. In the following, it will be summarized how the proposal helps achieving the initial requirements.

- *Location transparent addressing:* The architecture makes use of the existing agent based addressing, i.e. each stream has an initiator and participant agent identifier. An agent sending data to another agent can use the target's agent identifier to create a virtual stream connecting both without knowledge where this agent resides.
- *Infrastructure traversal:* The layered model allows for a clear separation of concerns and enables the streaming mechanism to utilize the existing FIPA transport scheme. As a result, if platforms manage to reach one another through some channel, it can be used simultaneously for standard messaging as well as for binary streaming. Stream data is automatically fragmented into small packets, which can be multiplexed with other data.
- *Failsafe connections and heterogeneous multihoming:* The message service uses multihoming by setting up different transports. Connections are virtual, using all transports available, i.e. the message service will try to send messages (binary as well as standard) via different transports until it fails and no alternatives are available.
- *Non-functional properties:* Streams can be initialized with non-functional properties. These are used by the connection and message service to handle the connections properly. In the current implementation only non-functional criteria for security related aspects are supported.
- *Agent integration:* Streaming is available at the agent level by streaming APIs. These completely relieve a developer from low-level communication issues. The integration supports the reactive nature of agents by using a non-blocking approach without additional threading within an agent.

Besides these aspects also performance of the streaming approach is an important factor for its usefulness in practice. Using different example applications the performance has been compared with the original performance of a direct TCP connection. The testing has revealed that the performance is very close to a direct connection so that the comfort of using the APIs does not lead to a trade-off decision between speed and usability.

As important part of future work we plan to add support for more non-functional aspects. In particular, we want to support stream priorities and unreliable streams suitable for audio and video transmission, where outstanding packets should be discarded and not resend.

References

1. Andersen, D., Balakrishnan, H., Kaashoek, F., Morris, R.: Resilient overlay networks. In: Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles, SOSP 2001, pp. 131–145. ACM, New York (2001)

2. Bellifemine, F., Bergenti, F., Caire, G., Poggi, A.: JADE - A Java Agent Development Framework. In: Multi-Agent Programming: Languages, Platforms and Applications, pp. 125–147. Springer (2005)
3. Bellifemine, F., Poggi, A., Rimassa, G.: Developing multi-agent systems with a fipa-compliant agent framework. *Softw., Pract. Exper.* 31(2), 103–128 (2001)
4. Bless, R., Mayer, C., Hübsch, C., Waldhorst, O.: SpoVNet: An Architecture for Easy Creation and Deployment of Service Overlays, pp. 23–47. River Publishers (June 2011)
5. Braubach, L., Pokahr, A.: Developing Distributed Systems with Active Components and Jadex. *Scalable Computing: Practice and Experience* 13(2), 3–24 (2012)
6. Foundation for Intelligent Physical Agents (FIPA). FIPA Abstract Architecture Specification, Document no. FIPA00001 (December 2002)
7. Sutter, H., Larus, J.: Software and the concurrency revolution. *ACM Queue* 3(7), 54–62 (2005)

Strategic Behaviour in Multi-Agent Systems Able to Perform Temporal Reasoning

Matei Popovici and Lorina Negreanu

Abstract. Temporal reasoning and strategic behaviour are important abilities of Multi-Agent Systems. We introduce a method suitable for modelling agents which can store and reason about the evolution of an environment, and which can reason strategically, that is, make a rational and self-interested choice, in an environment where all other agents will behave in the same way. We introduce a game-theoretic formal framework, and provide with a computational characterisation of our solution concepts, which suggests that our method can easily be put into practice.

1 Introduction

Multiagent Systems (MAS) have long been a successful means for modelling the interaction of software agents (or simply, programs) with themselves, other humans, and a given environment. In this context, there is an ever increasing need for MAS: (i) able to perform temporal inferences and (ii) capable of strategic reasoning.

In this paper, we equip MAS with memory and temporal inference capabilities, by deploying a method for temporal reasoning previously described in [2, 3, 6, 5], and we study the strategic behaviour of such systems. For the latter, we introduce the game-theoretic concept of Nash Equilibrium, and show how it can be used for enforcing MAS stability. We assume each agent has a particular goal, which is dependent on the current and past state(s) of the system. Goals are expressed using the temporal language $L_{\mathcal{H}}$, introduced and described in [5]. Each agent has some available actions which are able to change the system state. Also, each action has associated a particular cost. We further assume that agents are *rational* and *self-interested*, meaning they will choose to execute those actions which: (i) make the goal satisfied, (ii) minimises the agent's costs. We are interested in those situations

Matei Popovici · Lorina Negreanu
Computer Science Department, University Politehnica of Bucharest,
Bucharest, Romania
e-mail: matei.popovici@cs.pub.ro, lorina@moon.ro

where agents cannot individually satisfy their goals i.e. they may have (partially) conflicting and/or (partially) overlapping goals. In such situations, agents may deviate, i.e. change their set of actions to another, if the latter is a possible means for achieving a better outcome. The solution concept we use, the Nash Equilibrium, is aimed at identifying the outcomes where no agent has an incentive to deviate. We further study the computational complexity of related to the Nash Equilibrium, using the $L_{\mathcal{H}}$ -model checking procedure described in [5] and provide with an upper complexity bound. We conjecture that such a bound is tight.

Related Work: We consider Boolean Games [4] to be one of the first frameworks which use logic for describing goal-based strategic interactions between agents. However, instead of propositional logic, we use the more expressive language $L_{\mathcal{H}}$, which also allows expressing domain-dependent temporal properties.

The choice of $L_{\mathcal{H}}$ over well-known temporal logics such as LTL [7] or CTL [1], is motivated, on one hand, on the increased computational complexity of model checking, in the case of LTL [7] and on the reduced expressive power of CTL, with respect to $L_{\mathcal{H}}$ [5].

The rest of the paper is structured as follows. In Section 2 we make a brief account on the temporal knowledge representation and reasoning method which we use. In Section 3 we introduce our formal framework, define the Nash Equilibrium solution concept and describe our complexity results, and in Section 4 we conclude and identify possible future work.

2 Temporal Representation and Reasoning

In what follows, we succinctly describe the approach for temporal representation and reasoning which we adopt. For a more detailed presentation, see [5].

Temporal Graph. The history of a given MAS is stored using *labelled* temporal graphs. A labelled temporal graph (or simply temporal graph) consists of: (i) *hypernodes* which model discrete moments of time, (ii) *labelled action nodes* which model instantaneous events that change the state of the current environment and (iii) *labelled quality edges* which model time-dependent properties.

An example of a temporal graph is shown in Figure 1, where $h_i, i = \overline{1,3}$ are hypernodes designating three distinct moments of time, a, b, c, d, e, f and a_{end} are action nodes, each associated to a certain hypernode, (a, b) , (c, d) and (e, f) are quality edges denoting the time-dependent properties $On(ac)$, $On(h)$ and $On(v)$, respectively.

Formally, given a *vocabulary* $\sigma_A \cup \sigma_Q$ and $\sigma_A \cup \sigma_Q$ -structure \mathfrak{I} , over universe I (a set of individuals), a labelled temporal graph (or t-graph) is a structure $\mathcal{H} = \langle A, H, \mathcal{T}, E, \mathcal{L}_A, \mathcal{L}_Q \rangle$, where A is a set of *action nodes*, H is a set of hypernodes, $\mathcal{T} : A \rightarrow H$ is a onto (or surjective) function which assigns for each action node a , the hypernode $\mathcal{T}(a)$ when a occurs, $E \subseteq A^2$ is a *quality edge* relation, the function $\mathcal{L}_Q : E \rightarrow \mathcal{P}(\cup_{R \in \sigma_Q} R^I)$ **labels** each quality edge with a set of relation instances from \mathfrak{I} and the function $\mathcal{L}_A : A \rightarrow \mathcal{P}(\cup_{R \in \sigma_A} R^I)$ **labels** each action node with a set of action

labels from \mathfrak{I} . Given the quality edge $(a, b) \in E$, we say that action nodes a or b are the *constructor* or *destructor* nodes of (a, b) , respectively.

The language $L_{\mathcal{H}}$. We use the language $L_{\mathcal{H}}$ in order to express complex temporal relations between time-dependent properties, such as *has the air conditioner been opened in the same time a window has been opened* or *find all properties On(ac) which are destroyed precisely when a property Opened(x,y) is created*.

Due to limited space, we provide with a simplified syntax and semantics of $L_{\mathcal{H}}$. The original one(s) can be found in [5].

Let Vars be a set of variables, \mathfrak{I} be a $\sigma_Q \cup \sigma_A$ -structure and $\mathcal{H}_{\mathfrak{I}}$ be a labelled t-graph. The *syntax* of a (Q)-formula is recursively defined with respect to \mathfrak{I} , as follows. If $R \in \sigma_Q$ with $\text{arity}(R) = n$ and $\vec{t} \in (\text{Vars} \cup I)^n$, then $R(\vec{t})$ is an **atomic** Q -formula (or an atom). If ϕ is a (Q)-formula then (ϕ) is also a (Q)-formula. If ϕ, ψ are (Q)-formulae then $\phi \mathbf{b} \psi, \phi \neg \mathbf{b} \psi, \phi \mathbf{m} \psi, \phi \mathbf{a} \psi, \phi \neg \mathbf{a} \psi$ are also (Q)-formulae. **b** stands for *before* and $\neg \mathbf{b}$ for *not before*. Similarly, **m** stand for *meets* and **a** for *after*. The following are valid $L_{\mathcal{H}}$ formulae: (i) $\text{On}(x)$ (ii) $\text{On}(ac) \mathbf{b} \text{Opened}(x,y)$ (iii) $\text{On}(x) \mathbf{m} \text{Opened}(John,x)$. Formula (i) refers to all properties *On* associated to some individual x , which can occur at any time in the evolution of the domain. Formula (ii) refers to those properties *On(ac)* which occur before any relation *Opened*, which may enrol arbitrary individuals x, y . Similarly, formula (iii) refers to those properties *On(ac)* which end at the very moment when some relation *Opened(John,x)* is created.

$L_{\mathcal{H}}$ formulae are evaluated over *paths* from labelled temporal graphs. The evaluation is defined by the mapping $\|\cdot\|_{\mathcal{H}}^Q : L_{\mathcal{H}} \rightarrow 2^E$ which assigns for each formula $\phi \in L_{\mathcal{H}}$, a set of quality edges $\|\phi\|_{\mathcal{H}}^Q$ which satisfy it. In what follows, we omit the formal definition of the $L_{\mathcal{H}}$ semantics, and defer the interested reader to [5]. Finally, we note that, in the unrestricted case considered in [5], $L_{\mathcal{H}}$ -formulae can also express more general constraints, for instance between actions and properties (and not just properties, as considered here). Returning to the temporal graph, we have that: $\|\text{On}(x)\|_{\mathcal{H}}^Q = \{(a_1, a_2)\}$, since (a_1, a_2) is labelled with *On(ac)*, which unifies with *On(x)*. $\|\text{On}(ac) \mathbf{b} \text{Opened}(John,x)\|_{\mathcal{H}}^Q = \{(a_1, a_2)\}$ since (a_1, a_2) also occurs before quality edge (a_3, a_4) , which is labelled with *Opened(John,win)*. Finally, $\|\text{On}(x) \mathbf{m} \text{Opened}(John,x)\|_{\mathcal{H}}^Q = \emptyset$ since there is no individual $i \in I$, for which properties *On(i)* and *Opened(John,i)* exist in our labelled temporal graph.

Proposition 1 ([5]). Let \mathcal{H} be a temporal graph where temporal nodes H are equipped with a temporal order: $\langle H, < \rangle$. We designate by $L_{\mathcal{H}}^{\leq}$ the language $L_{\mathcal{H}}$ defined over such temporal graphs. Given a formula $\phi \in L_{\mathcal{H}}^{\leq}$ and a property Q , the decision problem $Q \in \|\phi\|_{\mathcal{H}}^Q$ is NP-complete.

3 Formal Setting

In what follows, we formally introduce the concept of Nash Equilibrium adjusted to our setting, as well as other supporting concepts.

Definition 1 (t-frame). Let \mathfrak{I} be a structure over vocabulary $\sigma_Q \cup \sigma_A$ and with universe I , $\mathcal{A}set = \{R_i(\bar{i}) : R_i \in \sigma_A, \bar{i} \in R_i^I\}$, and $Qset = \{R_i(\bar{i}) : R_i \in \sigma_Q, \bar{i} \in R_i^I\}$.

A **t-frame** is given by $\mathcal{F} = (N, Own, Ont, c, (\phi_n)_{n \in N}, (v_n)_{n \in N}, \langle H, < \rangle)$, where N is a finite set of agents, $Own : N \rightarrow 2^{\mathcal{A}set}$ is a function that assigns to each agent a subset of action *types* he can control. $Own(n)$ models the types of changes agent n can perform to the environment. The sets $Own(1) \dots Own(|N|)$ are a **partition** of $\mathcal{A}set$, $Ont \subseteq \mathcal{A}set \times Qset \times \mathcal{A}set$ indicates what are the labels of action nodes that can create and destroy quality edges with a certain label. In this sense, *Ont* can be interpreted as a (simple) *ontology*. *Ont* must take into account that each quality edge (and hence quality label) must be created/destroyed by a unique action node (hence action label), $c : \mathcal{A}set \rightarrow \mathbb{R}^{\geq 0}$ assigns a cost to each action label, $(\phi_n)_{n \in N}$ is a set of goals, one for each agent ($\phi_n \in L_H$ for $n \in N$), $(v_n)_{n \in N}$ a goal value for each agent, measuring the amount of resources each agent is willing to spend for achieving his goal, a finite and ordered set $\langle H, < \rangle$ of hypernodes.

Example 1 (t-frame). Let \mathcal{F} be a three-agent frame where h (home cinema), ac (air conditioner) and v (ventilation) are individuals and $\mathcal{A}set = \{turnOn(x) : x \in I\} \cup \{turnOff(x) : x \in I\}$ and $Qset = \{On(x) : x \in I\}$. Each agent 1, 2, 3 can turn on or off device ac, h, v , respectively. The costs of each action are fixed to 1. The goals of each agent are as follows: $\phi_1 = On(ac)$ **b** $On(h)$ (*the air conditioner must be started before the home cinema was started*), $\phi_2 = On(ac)$ (*at some point in time, the air conditioner should be started.*), $\phi_3 = On(v)$ **m** $On(h)$ (*the ventilation should be stopped precisely when the home cinema starts*). Finally, $v_n = 5$ for $n \in \{1, 2, 3\}$, and $\langle H, < \rangle$ is given by: $h_1 < h_2 < h_3$.

In what follows, we assume that \mathcal{F} is a **t-frame** and $h_{max} \in H$ is the *most recent* hypernode, i.e., $\nexists h' \in H$ such that $h_{max} < h'$: An **action** of a agent n is given by: $\langle A_n, \mathcal{L}_{A_n}, \mathcal{T}_n \rangle$ where A_n is a set of action nodes, $\mathcal{L}_{A_n} : A_n \rightarrow \mathcal{A}set$ is a labelling function and $\mathcal{T}_n : A_n \rightarrow H$ is a mapping of A_n on the set of hypernodes. We note that, in this paper, we have restricted \mathcal{L}_{A_n} , such that each action node receives a unique label. This restriction is for convenience only.

An **action profile** consists of a vector $(\langle A_n, \mathcal{L}_{A_n}, \mathcal{T}_n \rangle)_{n \in N}$ of actions, one for each agent in N . An action profile **induces** a labelled temporal graph $\langle A, H, \mathcal{T}, E, \mathcal{L}_A, \mathcal{L}_Q \rangle$ where: $A = (\bigcup_{n \in N} A_n) \cup a_{end}$, for each pair $a, b \in A$, if there exists a label $X \in Qset$ such that $(\mathcal{L}_A(a), X, \mathcal{L}_A(b)) \in Ont$, then create $(a, b) \in E$, and for each such X , $X \in \mathcal{L}_Q(a, b)$, and if there exists an action node $a \in A$ and **no** action node $b \in A$ such that $(\mathcal{L}_A(a), X, \mathcal{L}_A(b))$, then create $(a, a_{end}) \in E$, and for each such X , $X \in \mathcal{L}_Q(a, b)$. Finally, $\mathcal{T}(x) = \mathcal{T}_i(x)$ if $x \in A_i$ and $\mathcal{T}(x) = h_{max}$ if $x = a_{end}$ and $\mathcal{L}_A(x) = \{\mathcal{L}_{A_i}(x)\}$ if $x \in A_i$, for $i = 1 \dots |N|$.

Proposition 2. Given a labelled temporal graph \mathcal{H} , there is a **unique** action profile $a^* \in \times_{n \in N} \mathcal{A}ctions_n$ such that a^* induces \mathcal{H} .

Given a labelled t-graph \mathcal{H} and a formula ϕ_k of agent k , we say ϕ_k is **satisfied** in \mathcal{H} iff $\|\phi_k\|_{\mathcal{H}}^X \neq \emptyset$. Thus, a goal ϕ_k is satisfied in \mathcal{H} if there exists at least one quality edge in \mathcal{H} that belongs to $\|\phi_k\|_{\mathcal{H}}^Q$. A labelled temporal graph \mathcal{H} induces a **cost**

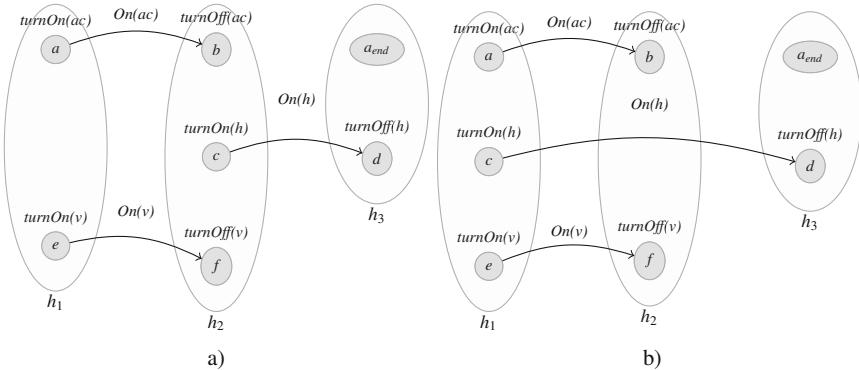


Fig. 1 a) A temporal graph induced from an action profile (and a Nash Equilibrium) and b) A temporal graph which is not a Nash Equilibrium

on a agent n , denoted $Cost_n(\mathcal{H})$, which is: $Cost_n(\mathcal{H}) = \sum_{a \in A} \left(\sum_{X \in Own(n) \cap \mathcal{L}_A(a)} c(X) \right)$. $Cost(n)$ is the sum of costs of actions performed by agent n . The utility $u_n(\mathcal{H})$ of agent n in a labelled temporal graph \mathcal{H} is defined as: $u_n(\mathcal{H}) = v_n - Cost_n(\mathcal{H})$ if ϕ_n is satisfied in \mathcal{H} , and as $u_n(\mathcal{H}) = -Cost_n(\mathcal{H})$, otherwise. We say a agent n prefers a labelled temporal graph \mathcal{H} over \mathcal{H}' and write $\mathcal{H} >_n \mathcal{H}'$, if $u_n(\mathcal{H}) > u_n(\mathcal{H}')$. We also extend the preference relation over action profiles. Given two action profiles a, a' that induce temporal graphs \mathcal{H} and \mathcal{H}' , respectively, we write $a >_n a'$ iff $\mathcal{H} >_n \mathcal{H}'$. Let \mathcal{H} be the labelled temporal graph from Figure 1 a) and \mathcal{H}' be the temporal graph from Figure 1 b). We have that $u_1(\mathcal{H}) = 5 - (1+1) = 3$ and $u_1(\mathcal{H}') = 0 - (1+1) = -2$, since agent 1's goal is satisfied in \mathcal{H} but not in \mathcal{H}' . Therefore $\mathcal{H} >_1 \mathcal{H}'$.

A **temporal game** is defined as $\mathcal{T}\mathcal{G} = (\mathcal{F}, (\mathcal{Actions}_n)_{n \in N}, (>_n)_{n \in N})$ where, for each $n \in N$, $\mathcal{Actions}_n$ designates the set of actions available to agent n and $>_n$ is the preference relation of agent n over temporal graphs. A **Nash Equilibrium** (NE) of a **temporal game** is a labelled temporal graph induced by an action profile: $(a_1^*, \dots, a_n^*, \dots, a_{|N|}^*) \in \times_{n \in N} \mathcal{Actions}_n$ such that, for each agent $n \in N$ and any action $a_n \in \mathcal{Actions}_n$: $(a_1^*, \dots, a_n^*, \dots, a_{|N|}^*) >_n (a_1^*, \dots, a_n, \dots, a_{|N|}^*)$. Nash Equilibria capture those situations in which each individual agent n cannot change his action (namely a_n^*) to one that ensures a higher utility. The t-graph in Figure 1 a) is a Nash Equilibrium. All three agents have their goals satisfied, and no individual agent can change his action to achieve a higher utility. The temporal graph from Figure 1 b) is not a Nash Equilibrium. The goal of agent 1 is not satisfied and there is no action that 1 can take in order to change this. Therefore, 1 would prefer not to execute any action node(i.e., remain passive), and therefore achieve 0 utility (instead of -2). The same is true with respect to agent 3.

Proposition 3 (Complexity Results). *Checking whether a temporal graph \mathcal{H} is a Nash Equilibrium of a temporal game $\mathcal{T}\mathcal{G}$ with goals formulated in the language $L_{\mathcal{TP}}^<$ is a Π_2 -problem. Finding if there exists a temporal graph \mathcal{H} such that \mathcal{H} is a Nash Equilibrium of a temporal game $\mathcal{T}\mathcal{G}$ with goals formulated in the language $L_{\mathcal{TP}}^<$ is a problem in Σ_3 .*

4 Conclusions and Future Work

While the scope of our paper is rather formal, we believe our results can be easily put into practice. Our complexity results show that implementations are possible, even if they require an increased computational effort. We consider this effort to be tolerable. Also, by introducing *limited memory*, that is, by truncating temporal graphs to a fixed number of hypernodes, the computational effort may be further controlled. As suggested by all the examples above, our modelling method can be used for identifying stable behaviour of agents in intelligent environments (i.e., buildings equipped with programmable sensors and control devices). Such an approach is currently a work-in-progress. However, we consider our method to be general enough for application in a wide variety of scenarios that require modelling temporal reasoning together with strategic behaviour.

Acknowledgements. The authors wish to acknowledge the help and support from prof. Cristian Giumale, which was the first to think about temporal graphs, and whose guidance patroned the work presented in this paper.

The work has been funded by Project 264207, ERRIC-Empowering Romanian Research on Intelligent Information Technologies/FP7-REGPOT-2010-1.

References

1. Arnold, A., Crubille, P.: A linear algorithm to solve fixed-point equations on transition systems. *Inf. Process. Lett.* 29(2), 57–66 (1988)
2. Giumale, C., Negreanu, L.: Reasoning with fluid qualities. In: 17th International Conference on Control Systems and Computer Science, CSCS-17, vol. 2, pp. 197–203 (December 2009)
3. Giule, C., Negreanu, L., Muraru, M., Popovici, M.: Modeling ontologies for time-dependent applications. In: International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, pp. 202–208 (2010)
4. Harrenstein, P., van der Hoek, W., Meyer, J.-J., Witteveen, C.: Boolean games. In: Proceedings of the 8th Conference on Theoretical Aspects of Rationality and Knowledge, TARK 2001, pp. 287–298. Morgan Kaufmann Publishers Inc., San Francisco (2001)
5. Popovici, M.: Using evolution graphs for describing topology-aware prediction models in large clusters. In: Fisher, M., van der Torre, L., Dastani, M., Governatori, G. (eds.) CLIMA XIII 2012. LNCS, vol. 7486, pp. 94–109. Springer, Heidelberg (2012)
6. Popovici, M., Muraru, M., Agache, A., Giumale, C., Negreanu, L., Dobre, C.: A modeling method and declarative language for temporal reasoning based on fluid qualities. In: Andrews, S., Polovina, S., Hill, R., Akhgar, B. (eds.) ICCS-ConceptStruct 2011. LNCS, vol. 6828, pp. 215–228. Springer, Heidelberg (2011)
7. Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. *J. ACM* 32(3), 733–749 (1985)

Control of a Mobile Robot by Human Gestures

Stefan-Gheorghe Pentiuc, Oana Mihaela Vultur, and Andrei Ciupu

Abstract. The control of a mobile robot based on natural human gestures represents a challenge and a necessity. In this paper we will try to present an innovative system that controls a mobile robot using human gestures performed with arms. The human operator acts in the front of a Microsoft Kinect sensor [1] that identifies the skeleton and transmits the coordinates of its joint points to the PC. It follows the gestures recognition process, and transmission to the mobile robot. We succeed to integrate the skeletal tracking, gesture recognition and the control of the mobile robot in an unique application. With this application a number of experiments were deployed, that permit us to evaluate the performances of the proposed interaction system. The overall system evaluation has been made using the following performance parameters: classification accuracy, error rate, precision, recall, sensitivity and specificity.

1 Introduction

Gestures are often used to accompany speech. They can make words more expressive. Nowadays we can use gestures not only to accompany speech, but also to navigate in a virtual environment [2], to simulate assembly operation inside virtual environments [3], or to control a device from real world, like a mobile robot, in our case.

Lately, several HRI based systems have been developed. M. Hahn et al. propose in [4] a HRI based system to control an industrial robot using hand gestures. They used Levenshtein Distance on Trajectories (LDT distance) for trajectories matching and Hidden Markov Models (HMMs) for recognizing different action sequences.

Stefan-Gheorghe Pentiuc · Oana Mihaela Vultur · Andrei Ciupu

Stefan cel Mare University of Suceava,
720229, Romania

e-mail: pentiuc@eed.usv.ro, vultur_oana@usv.ro,
aciupu@stud.usv.ro

P. Chang-Beom et al. propose in [5] a gesture recognition algorithm which allows a natural human robot interaction. Hands detection is made with the color distribution of the face region. The face was detected with an improved version of the Viola-Jones detector. Tracking was realized using a 3D particle filter and the estimation of the pointing direction is based on a cascade of HMMs.

M. Van den Bergh et al. presents in [6] a functional HRI based system which uses a real-time 3D hand gestures recognition algorithm. They propose a human detection method which includes skin, face and leg detection.

T. Cerlinca et al. propose in [7] a system that controls an industrial robot in a very natural way, through 3D hand gestures. The algorithm used for gesture recognition was Dynamic Time Wrapping.

The paper will present a new technique for the control of a mobile robot using dynamic gestures performed with arms. We made a performance analysis of our interaction system.

As issues of the experimentation stage are evaluated the following performance parameters: classification accuracy of the system, error rate, precision, recall, sensitivity, and specificity.

The paper is organized in 6 sections as follows: Introduction, Hardware and software architecture of the interaction system, Models and algorithms for gesture recognition, Gesture set used for interaction and application functionality, Experimental results, and Conclusions.

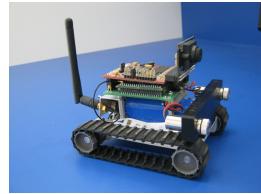
2 Interaction System Hardware and Software Architecture

2.1 *Hardware Architecture*

The application runs on a PC with Intel(R) Core (TM) 2 Quad Q9650 processor at 3GHz frequency and 4GB RAM. We used a Kinect sensor [1] and a mobile robot (Surveyor's SRV-1) [8]. The Kinect sensor is a device who has a small base connected to a motorized pivot. Kinect sensor has an RGB camera, two depth sensors and multi-array microphone. It has some capabilities to provide full-body 3D motion capture. The Kinect sensor provides an image matrix, and a depth matrix with z-coordinate for every pixel.

The mobile robot used for interaction is Surveyor's SRV-1. Designed for research, education, and exploration, Surveyor's SRV-1 internet-controlled robot employs the SRV-1 Blackfin Camera Board with 1000MIPS 500MHz Analog Devices Blackfin BF537 processor, a digital video camera with resolution from 160x128 to 1280x1024 pixels, laser pointer or optional ultrasonic ranging, and WLAN 802.11b/g networking on a quad-motor tracked mobile robotic base [8].

Fig. 1 Surveyor's SRV-1
mobile robot



2.2 Software Architecture

Software architecture consists of Operating System, Microsoft Visual Studio 2010 Professional, Kinect SDK, software application responsible for images acquisition, gestures recognition and transmission of commands to the robot.

Microsoft Kinect SDK (Software Development Kit) includes Windows 7 compatible PC drivers for Kinect sensor. Microsoft Kinect SDK provides Kinect capabilities for developers to build software applications with C++, C# or Visual Basic using Microsoft Visual Studio 2010.

This SDK includes skeletal identification and tracking, raw sensor streams.

In our system we used Microsoft Kinect SDK to develop an application that performs image acquisition and gestures recognition (using Dynamic Time Warping algorithm [9]). After a gesture representing a specific command is recognized, a correspondent command is transmitted to the robot. For example, if the recognized gesture is “advance” - we transmit to the IP address of the robot the ‘8’ command, if the gesture is recognized as “behind” gesture - we transmit to robot the ‘2’ command, if is “left” - we transmit to the IP address of the robot the ‘4’ command, if the gesture is recognized as “right” - we transmit to robot the ‘6’ command.

3 Models and Algorithms for Gesture Recognition

The system of gesture recognition is based on a previous learning stage. Learning or training stage consists of building a training set containing the description of all the gestures used for control, performed by various users. Each gesture description is called a pattern, and is analyzed and labeled by a human expert. The patterns in the training set are composed by skeleton configurations at different moments of time. A skeleton configuration is defined by the coordinates of the six joints. These six points of interest from the skeleton are: wrist right, elbow right, shoulder right, wrist left, elbow left, shoulder left. At every change of the skeleton position an event is generated, and the coordinates corresponding to the six points of interest are stored by the application program in a file.

In the recognition process the new gestures are compared with sequence of gestures from the training set using the DTW (Dynamic Time Warping) algorithm. This algorithm calculates DTW minimum distance between two coordinated sequences: model sequence and candidate sequence (to be a gesture). DTW algorithm finds the best match between the two sequences.



Fig. 2 The gestures set: a) “advance” (this gesture commands the mobile robot to go forward) b) “behind” (this gesture commands the mobile robot to go back) c) “right” (commands the mobile robot to go to the right) d) “left” (commands the mobile robot to go to the left)

4 Gesture Set

We used four gestures to control the mobile robot. The gesture set is presented in figure 2. The gestures are: “advance”, “behind”, “left” and “right”. The gesture acquisition is performed by the Kinect sensor. The recognition process is based on the training set and gestures are recognized using DTW algorithm.

Gestural interface provides to users a simple and intuitive interaction to control a robot using Kinect sensor. Commands for the robot are performed with the right or the left hand by indicating the direction forward, behind, left and right. All commands from the host (computer + kinect) to the SRV-1 robot consist of ASCII characters or decimal ASCII characters. All orders are confirmed by a robot to acknowledge the host as a “#” followed by a command. All commands can be executed using a program with TCP / telnet communications capability.

5 Experimental Results and Discussion

We tested the system using a total of 400 gestures. We run each classifier (“advance”, “behind”, “left” and “right”) and we performed, for each classifier, 100 gestures of “advance”, 100 gestures of “behind”, 100 gestures of “right” and 100 gestures of “left”.

After performing tests we obtained the confusion matrix [10] shown in table 1.

Table 1 The confusion matrix

Classes	Advance	Behind	Left	Right
TP	63	64	66	53
TN	271	300	296	300
FP	29	0	4	0
FN	37	36	34	47

System evaluation was performed using the following performance parameters: classification accuracy, error rate, precision, recall, sensitivity (or true positive rate) and specificity (or true negative rate).

A classified pattern is considered a sample that may be either positive or negative [11]. The classifier decision is represented in a structure known as a confusion matrix [11]. The confusion matrix has four categories: true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) [11]. True positives (TP) are samples correctly identified as positives. True negatives (TN) correspond to negatives samples correctly identified as negative. False positive (FP) refers to negative samples incorrectly identified as positives. False negatives (FN) refers to positive samples incorrectly identified as negative [11]. From the contents of this confusion matrix, some performance parameters can be calculated, such as: classification accuracy, error rate, precision, recall, sensitivity and specificity.

Classification accuracy, error rate, precision, recall, sensitivity and specificity obtained for every gesture are presented in table 2.

Table 2 Classification accuracy, error rate, precision, recall, sensitivity and specificity

Gesture	Accuracy	Error rate	Precision	Recall	Sensitivity	Specificity
Advance	84%	17%	68%	63%	63%	90%
Behind	91%	9%	100%	64%	64%	100%
Left	91%	10%	94%	66%	66%	99%
Right	88%	12%	100%	53%	53%	100%

We obtained an average accuracy rate of about 88.5% with the maximum value for “behind” and “left” gestures: 91%. The average precision rate obtained is 90.5% with the maximum value for “behind” and “right” gestures: 100%. We obtained an average sensitivity of about 61.5% with the maximum value for “left” gesture: 66% and an average specificity of about 97.25% with the maximum value for “behind” and “right” gestures: 100%.

6 Conclusions

In this paper we presented a new technique of interaction with a mobile robot using dynamic gestures performed with arms and we made a performance analysis of the interaction system. We calculated the following performance parameters: classification accuracy of the system, error rate, precision, recall, sensitivity and specificity.

The main contributions of this work are the new interaction techniques proposed (using dynamic gestures performed with arms) and the performance analysis of our interaction system.

Acknowledgements. This paper is supported in part by the project “Doctoral Burses at USV” – Contract no. POSDRU/6/1.5/S/22 project co-funded from European Social Fund through Sectoral Operational Program Human Resources 2007-2013.

References

1. Microsoft Kinect for Windows SDK beta – Programming Guide
2. Vultur, O., Pentiuc, S.G., Ciupu, A.: Navigation System in a Virtual Environment by Gestures. In: Proceedings of the IEEE 9th International Conference on Communications (COMM), Bucharest, Romania (2012)
3. Craciun, E.G., Grisoni, L., Pentiuc, S.G., Rusu, I.: Novel Interface for Simulation of Assembly Operations in Virtual Environments. In: Advances in Electrical and Computer Engineering, pp. 47–52 (2013),
<http://dx.doi.org/10.4316/AECE.2013.01008>
4. Hahn, M., Kruger, L., Wohler, C., Kummert, F.: 3D Action Recognition in an Industrial Environment. In: Proceedings of the 3rd International Workshop on Human-Centered Robotic Systems (HCRS 2009), Bielefeld, Germany (2009)
5. Chang-Beom, P., Myung-Cheol, R.R., Seong-Whan, L.: Real-time 3D pointing gesture recognition in mobile space. In: Proceedings of the 8th IEEE International Conference on Automatic Face & Gesture Recognition (2008)
6. Van den Bergh, M., Carton, D., De Nijls, R., Mitsou, N., Landsiedel, C., et al.: Real-time 3D hand gesture interaction with a robot for understanding directions from humans. In: Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (2011)
7. Cerlinca, T., Pentiuc, S.G., Vlad, V.: Real-time 3D Hand Gestures Recognition for Manipulation of Industrial Robots. In: Elektronika ir Elektrotechnika, pp. 3–8 (2013)
8. http://www.surveyor.com/SRV_info.html
9. Senin, P.: Dynamic Time Warping Algorithm Review, pp. 1–23. University of Hawaii al Manoa (2008)
10. Khati, P.: Comparative Analysis of Protein Classification Methods (2004)
11. Falinouss, P.: Stock Trend Prediction Using News Articles. A Text Mining Approach (2007), <http://epubl.ltu.se/1653-0187/2007/071/LTU-PB-EX-07071-SE.pdf>

Improving Noise Robustness of Speech Emotion Recognition System

Łukasz Juszkiewicz

Abstract. In this paper method of improving noise robustness of speech emotion recognition system is proposed. Such a system has been developed for use in a social robot, but its performance is highly degraded by environmental noise. To alleviate this problem, the histogram equalisation is proposed to reduce the difference between feature vectors in clean and noisy conditions. In training phase of the system the averaged histograms of pitch and MFCC are computed and then serve as reference for equalisation. System performance was evaluated using Database of Polish Emotional Speech, which was split into training and test sets. Test sets were noised with 3 different noise samples. Presented preliminary results show a significant improvement of recognition accuracy in noisy environment conditions.

1 Introduction

Emotion recognition is a crucial task in social robotics. In order to communicate in a natural way with human, the robot should be able to recognise different expressions of emotions: gestures, facial expressions, content of speech and the tone of voice. This work focuses on the last issue, i. e. the recognition of emotions encoded in the human voice intonation.

Social robots are often intended as daily companions of people. Therefore the speech emotion recognition system of such robot should be able to operate correctly also in noisy places, like for example public buildings. Changeable acoustic conditions, ambient noise, sounds of motors of the robot etc. cause severe mismatch between speech parameters in the training and the evaluation phase of the system operation, which evokes degradation in the recognition accuracy.

Łukasz Juszkiewicz
Wrocław University of Technology
Institute of Computer Engineering, Control and Robotics
ul. Janiszewskiego 11/17 50-370 Wrocław
e-mail: lukasz.juszkiewicz@pwr.wroc.pl

In the field of automatic speech recognition, several methods were developed to address this problem [6]. They fall into three main categories:

- removing noise from the speech signal,
- extracting noise-robust features,
- adapting the acoustic model.

These techniques were adapted for robustifying speech emotion recognition systems. Several different approaches to this problem could be found in the literature: finding more noise-robust parameters of speech [7], extracting large features sets [17], speech enhancement by adaptive noise cancellation [21], adaptive acoustic space adaptation [19]. This paper concentrates on the feature normalisation for improving their robustness. The histogram equalisation technique is proposed to reduce the difference between feature vectors in clean and noisy conditions. Described system was robustified and its performance was evaluated using the emotional speech corpora.

The intended application of the system is being part of software of the robot FLASH (see Fig. 1) [2]. FLASH (Flexible LIREC Autonomous Social Helper) is a mobile-dexterous-social robot designed and constructed by Wrocław University of Technology as a contribution to EU project LIREC. It is based on two-wheel balancing platform carrying its torso, which supports other components: the EMYS head and arms. EMYS (EMotive headY System) possesses eleven degrees of freedom and its face can display 6 basic emotions such as surprise, disgust, fear, angeriness, joy and sadness. Two arms have 7 degrees of freedom each and their structure resembles human arms. They are equipped with dexterous hands WANDA (Wrut hANDs for gesticulAtion). FLASH is used for human-robot interactions experiments and emotion recognition is one of the crucial competences of such robot.



Fig. 1 Social robot FLASH

The rest of the paper is organised as follows. In section 2 the structure of the discussed system is introduced. In section 3 modification of system is described. In section 4 preliminary evaluation results are discussed. Finally, section 5 presents conclusions and further research.

2 Structure of the Existing System

Commonly used pattern recognition algorithms are applicable to the speech emotion recognition. However, there are at least two different approaches. One is estimating the short-time signal parameters and modelling their changes with Hidden Markov Models or similar tools [14, 12]. The other is extracting global features of the signal and applying statistical methods and various types of classifiers: SVM [24, 4], artificial neural networks [23, 5], decision trees [20], LDA [13]. The second approach was chosen—each utterance is analysed as a whole, so global features are extracted and then used for classification.

The speech emotion recognition system [10] developed by this author has the form of a set of MATLAB scripts using external tools — the programs Praat and Weka. Praat is a free (Open Source) program for phonetic analysis of speech. It can compute several parameters of speech: pitch, formants, spectrum, mel-frequency cepstral coefficients and many others [15]. Weka 3 (Waikato Environment for Knowledge Analysis) is a popular suite of machine learning software written in Java. It contains a collection of visualization tools and algorithms for data preprocessing, filtration, clasterisation, classification, feature selection and regression modelling [8]. It is freely available under GNU General Public License. The structure of the system is illustrated in figure 2. There are two phases of the system's operation: learning phase and evaluation phase. In the off-line learning phase feature selection and supervised learning of classifier is carried. In the evaluation phase the prepared system is used for on-line speech emotion classification.

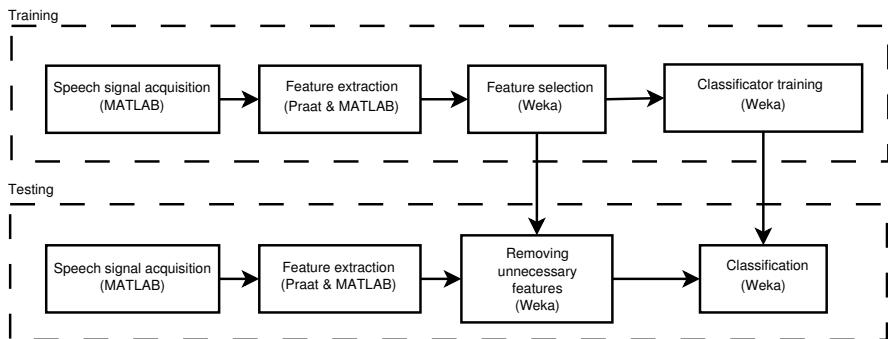


Fig. 2 Block diagram of speech emotion recognition system

The speech signal is parametrised by Praat. The following parameters are computed:

- pitch (fundamental frequency),
- mel-frequency cepstral coefficients (MFCC),
- harmonic to noise ration (HNR),
- long-term averaged spectrum (LTAS),
- intensity,
- spectrogram.

In order to extract more useful information from obtained vectors additional vectors are derived from them:

- first and second order difference,
- values of local minima,
- values of local maxima,
- distance between adjacent extrema,
- value difference of adjacent extrema,
- slopes between adjacent local extrema,
- absolute values of the two above.

These parameters are time-series — their length depends on the duration of analysed utterance. For classification purposes, it is necessary to convert the time series into a feature vector of fixed length. This is achieved by treating time series as outcomes of random variables and computing their statistics:

- arithmetic mean,
- median,
- standard deviation,
- global maximum,
- global minimum,
- first quartile,
- second quartile,
- range,
- interquartile range.

Using this method 1722 features are generated. The structure of the feature vector is illustrated in figure 3. The number of features is then reduced in the feature selection process.

Feature selection is done with algorithm provided by Weka, which aim is to find the subset of features that are highly correlated with class, while having low inter-correlation. Classifier is a Bayes net, also borrowed form Weka package, using a simple probability estimator and a hill climbing search algorithm.

The main problem in applying the system to social robot such as FLASH is noise: sounds generated by the robot itself and environmental noise. It is difficult to completely filter out those noises so robustness of system to noise is highly desired. Unfortunately, performance of the existing system downgrades considerable, if analysed recordings are noised. Recognition accuracy of 4 emotions (neutral, joy,

sadness and anger) from Database of Polish Emotional Speech (see sec. 4.1) was 81%. Then zero mean white noise was added to recordings from the test set to archive 30dB signal to noise ratio. As a result the recognition accuracy fell down to 25%, which means no recognition at all. With different types of noise results were very similar.

3 Improving Noise Robustness

Additive noise introduces a nonlinear distortion to the speech signal parameters space that has to be compensated. Several methods could be used for this purpose. They can be categorised as model-based and data distribution based. In the first technique statistics of the signal (mean, variance etc.) are normalised. Those methods are Cepstral Mean Normalisation (CMN), Mean and Variance Normalisation (MVN) and derived from them such as Short-Time MVN and other [22]. The data distribution based methods try to normalise a signal distribution to the reference. Short-time Gaussianisation (STG) [3] and Histogram Equalisation (HEQ) [9] methods fall into this category.

The histogram equalisation method utilises a certain property of random variables [16]. Suppose that X is a random variable with cumulative density function $F(x)$ and probability density function $f(x)$. X could be transformed into a random variable Y with reference probability density function $f_r(y)$ and cumulative density function $F_r(y) = F(x)$ using an invertible transformation $y = T(x)$. Equality of cumulative density functions

$$F_r(y) = F_r(T(x)) = F(x) \quad (1)$$

leads to the expression of the transformation $T(x)$:

$$T(x) = F_r^{-1}(F(x)). \quad (2)$$

It can be shown, that if the nonlinear distortion of the speech signal parameters introduced by additive noise is an invertible transformation, than the HEQ normalised parameters vector of clean utterance and the normalised parameters vector of noised utterance should be equal. Let x be parameter vector that is subject of an invertible

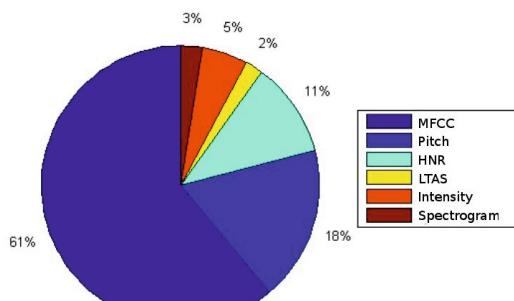


Fig. 3 Structure of feature vector

transformation $y = G(x)$ caused by noise. Then the cumulative density functions will be

$$F_x(x) = F_y(N(x)) \quad (3)$$

and as a result, the normalised vectors x_n and y_n will also be equal:

$$y_n = F_r^{-1}(F_y(N(x))) = F_r^{-1}(F_x(x)) = x_n. \quad (4)$$

The histogram equalisation (HEQ) of MFCC coefficients is widely used in a noise robust speech recognition [6]. This was the main motivation for making in this paper an attempt to use HEQ for improving the speech emotion recognition accuracy in noisy environment conditions.

3.1 Structure of Robustified System

The speech emotion recognition system has been modified to include the histogram equalisation technique for improving noise robustness. The MATLAB Image Processing Toolbox provides a histogram equalisation function `histeq` [1], which chooses the transformation $T(x)$ to minimise

$$|C_1(T(x)) - C_0(x)|, \quad (5)$$

where C_0 is the cumulative histogram of signal and C_1 is the cumulative sum of the reference histogram. After estimating speech signal parameters, histograms of pitch and each of the MFCC coefficients are equalised with respect to reference histograms. Those are histograms of averaged pitch and MFCCs of recordings from a training set, which are assumed to be not noised. The histogram equalisation is done also in the training phase to keep consistency between feature vectors in the training and the application phases. The feature extraction and selection stages were not changed neither the classification stage. Structure of modified system is shown in figure 4.

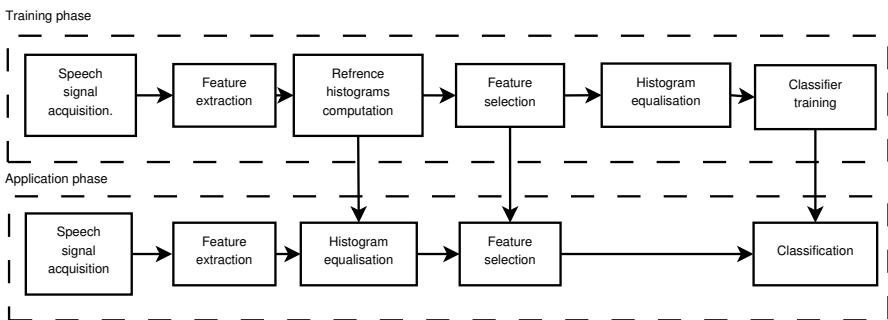


Fig. 4 Structure of robustified speech emotion recognition system

4 Results

The performance of the robustified system was evaluated using emotional speech corpora. Training and test sets were prepared. Test sets were noised to find out how it would affect the recognition accuracy.

4.1 Emotional Speech Corpora

The database of Polish Emotional Speech consists of 240 recordings [11]. Four male and four female actor speakers were asked to enact in five phrases five different emotions (anger, joy, sadness, fear and boredom) as well as the emotionally neutral state. Phrases are uttered in Polish and their meaning is emotionally neutral. They were recorded in a professional studio, so the sound quality is very high and the noise level is very low. The number of recordings is the same for each emotion. These recordings were evaluated by 50 humans — each of them was asked to classify 60 randomly selected samples. The average recognition ratio was 72%.

4.2 Test Sets

From 160 recordings 112 were randomly chosen (with stratification — number of instances in each class was the same) to form a training set. Other 48 were used to make noised test sets. Recordings were mixed, by program `sox`, with samples of different noises: street traffic sounds, sound of about hundred people speaking in a canteen and sound of a vacuum cleaner. For each type of noise test sets with signal to noise ratio of 30dB, 20dB, 10dB and 0dB were produced. Recordings from the training set were not modified.

4.3 Results of Tests

Training and test sets were prepared as described in section 4.2 — the procedure was repeated 5 times to get 5 training sets and 5 test sets for each type of noise and SNR combination. Differently from the other works [18], our system was trained using only clean recordings. Selecting features and training the classifier separately for each type of noise and its dB level combination would lead to the better recognition accuracy for this particular combination, but is completely unfeasible in real application. Tables 1 and 2 present the results of the system evaluation for different types of noise and signal to noise ratios. Presented values are averaged recognition accuracies. Infinite SNR means not noised recordings.

Table 1 Recognition accuracy of orginal system

SNR	∞	30dB	20dB	10dB	0dB
Accuracy (any noise)	81%	25%	25%	25%	25%

Table 2 Recognition accuracy of robustified system

SNR	∞	30dB	20dB	10dB	0dB
Accuracy (talks)	78%	75%	73%	60%	47%
Accuracy (street)	76%	68%	60%	43%	35%
Accuracy (vac. cl.)	77%	55%	43%	33%	33%

Results show a significant improvement in recognition accuracy in the presence of noise as well as slightly lower recognition accuracy in clean conditions, comparing with the original system. However improvement depends on type of noise i. e. its frequency spectrum, dynamic range and other parameters. Best results were achieved for sounds of talking people, slightly worse for street sounds and the worst for vacuum cleaner noise — distortion introduced by certain types of noise cannot be treated any longer as the invertible transformation of the parameters space and compensated by the histogram equalisation. Moreover, in case of very low SNR, it should be noted that even humans might have problems with recognising correctly the emotion of recorded speech — emotional content is effaced by the noise. Lower classification accuracy in clean conditions is a side effect of the histogram equalisation. MFCC and pitch histograms depend also on the emotion of speech. In the original system their variations have a contribution in the recognition process. After the histogram equalisation some of this information is lost, even though invertible transformation is used for equalisation.

5 Conclusions

This paper presents a robustification of the speech emotion recognition system dedicated to the social robot FLASH. Histogram equalisation technique was used to compensate nonlinear distortion of speech parameters (MFCC and pitch) introduced by noise. Reference histograms were computed as averaged histograms of the training set utterances. Preliminary evaluation of the robustified system performance shows that recognition accuracy highly depends on type of noise. In case of corruption by sounds of talking people system performs well even at moderate SNR ratios, but for a vacuum cleaner noise results are much worse. Experiments with larger data sets should be conducted to verify those results. Future research aims at speaker dependent recognition and evaluating real-time recognition performance of the system.

References

1. Enhance contrast using histogram equalization,
<http://www.mathworks.com/help/images/ref/histeq.html>
2. Robot FLASH, <http://lirec.ict.pwr.wroc.pl>

3. Alam, M.J., Ouellet, P., Kenny, P., O'Shaughnessy, D.: Comparative evaluation of feature normalization techniques for speaker verification. In: Travieso-González, C.M., Alonso-Hernández, J.B. (eds.) NOLISP 2011. LNCS (LNAI), vol. 7015, pp. 246–253. Springer, Heidelberg (2011)
4. Casale, S., Russo, A., Scebba, G., Serrano, S.: Speech emotion classification using machine learning algorithms. In: Proceedings of the 2008 IEEE International Conference on Semantic Computing, pp. 158–165. IEEE Computer Society, Washington, DC (2008), doi:10.1109/ICSC.2008.43
5. Dautenhahn, K.: Creating emotion recognition agents for speech signal. Multiagent systems, artificial societies, and simulated organizations. In: Socially Intelligent Agents: Creating Relationships with Computers and Robots. Kluwer Academic Publishers (2002)
6. García, L., Segura, J.C., de la Torre, Á., Benítez, C., Rubio, A.: Histogram equalization for robust speech recognition (2008),
http://www.intechopen.com/books/speech_recognition/histogram_equalization_for_robust_speech_recognition
7. Georgogiannis, A., Digalakis, V.: Speech emotion recognition using non-linear teager energy based features in noisy environments. In: 2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO), pp. 2045–2049 (2012)
8. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explor. Newsl. 11, 10–18 (2009), doi:<http://doi.acm.org/10.1145/1656274.1656278>
9. Hilger, F., Ney, H.: Quantile based histogram equalization for noise robust large vocabulary speech recognition. IEEE Transactions on Audio, Speech, and Language Processing 14(3), 845–854 (2006)
10. Juszkiewicz, Ł.: Speech emotion recognition system for social robots. In: Postępy Robotyki, pp. 695–704. Oficyna wydawnicza PW (2012) (in Polish)
11. Lodz University of Technology, Medical Electronics Division: Database of Polish Emotional Speech, http://www.eletele.p.lodz.pl/bronakowski/med_catalog/docs/licence.txt
12. Mao, X., Zhang, B., Luo, Y.: Speech emotion recognition based on a hybrid of HMM/ANN. In: Proceedings of the 7th Conference on 7th WSEAS International Conference on Applied Informatics and Communications, vol. 7, pp. 367–370. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point (2007)
13. Neiberg, D., Elenius, K.: Automatic recognition of anger in spontaneous speech
14. Nwe, T.L., Foo, S.W., Silva, L.C.D.: Speech emotion recognition using hidden markov models. Speech Communication 41, 603–623 (2003), doi:10.1016/S0167-6393(03)00099-2
15. Paul Boersma, D.W.: Praat: doing phonetics by computer, version 5.2.05 (2010), <http://www.praat.org/>
16. Peebles, P.Z.: Probability, random variables, and random signal principles / Peyton Z. Peebles Jr., 3rd edn. McGraw-Hill, New York (1993)
17. Schuller, B., Arsic, D., Wallhoff, F., Rigoll, G.: Emotion recognition in the noise applying large acoustic feature sets. In: Speech Prosody (2006)
18. Schuller, B., Seppi, D., Batliner, A., Maier, A., Steidl, S.: Towards more reality in the recognition of emotional speech. In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2007, vol. 4, pp. IV-941–IV-944 (April 2007)
19. Schuller, B.W.: Speaker, noise, and acoustic space adaptation for emotion recognition in the automotive environment. In: 2008 ITG Conference on Voice Communication (SprachKommunikation), pp. 1–4 (2008)

20. Sidorova, J.: Speech emotion recognition with TGI+.2 classifier. In: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop, EACL 2009, pp. 54–60. Association for Computational Linguistics (2009)
21. Tawari, A., Trivedi, M.: Speech emotion analysis in noisy real-world environment. In: 2010 20th International Conference on Pattern Recognition (ICPR), pp. 4605–4608 (2010)
22. Viikki, O., Laurila, K.: Cepstral domain segmental feature vector normalization for noise robust speech recognition. *Speech Communication* 25(1-3), 133–147 (1998), doi:10.1016/S0167-6393(98)00033-8
23. Xiao, Z., Dellandréa, E., Dou, W., Chen, L.: Hierarchical Classification of Emotional Speech. Tech. Rep. RR-LIRIS-2007-006, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon (2007)
24. Zhou, J., Wang, G., Yang, Y., Chen, P.: Speech emotion recognition based on rough set and SVM. In: Yao, Y., Shi, Z., Wang, Y., Kinsner, W. (eds.) IEEE ICCI, pp. 53–61. IEEE (2006)

Developing an Avatar Model for Driving Simulators

Razvan-Vlad Vasiu, Ligia Munteanu, and Cornel Brisan

Abstract. Driving a vehicle in safety conditions means actually to maintain an equilibrium in a complex system in which the main components are the runway, the vehicle and the driver. The complexity of the situations that could appear, the safety aspects and the continuous technological innovations in automotive industry have imposed the development of technical systems capable in simulating the vehicle driving, i.e. driving simulators. The dynamic pace of automotive development and the increase of safety standards have led to the need to continuously upgrading such devices. In order to reduce development time and resources, a virtual driving simulator is described that allows the user to interact with a computer based simulation (avatar) version, before having to build a full-scale simulator. Thus, the main goal of the current paper is to describe the concept of the avatar driving simulator. This avatar version of a driving simulator, named Computer Avatar Versatile Driving Simulator - (CAV-Drive) is considered to be a computerized model that allows the implementation in a software environment (for example, Matlab) different runway models, vehicles - suspensions, chassis - and driving behaviours for testing and optimisation in the virtual world different types of driving simulators.

1 Aspects for an Avatar Driving Simulator Model

1.1 The Concept of Avatar Driving Simulator

The current paper proposes a new concept to develop and analyse a driving simulator. This new concept is based on the correlation between the real world entities

Razvan-Vlad Vasiu · Cornel Brisan
Technical University of Cluj-Napoca,
B-dul Muncii nr.103-105, cod 400641, Cluj-Napoca, Romania
e-mail: {razvan.vasiu,cornel.brisan}@mdm.utcluj.ro

Ligia Munteanu
Institute of Solid Mechanics of Romanian Academy, Bucharest, Romania
e-mail: ligia_munteanu@hotmail.com

that generate cues for the human subjects when driving and their graphical representation (avatar) counterparts which simulate the corresponding cues in an artificial simulated environment (fig. 1). Besides the driver, the physical entities that appear in the real world are:

- the vehicle - suspensions and the direction mechanisms
- the runway (road) - along which the vehicle is moving with respect to its geometry and roughness

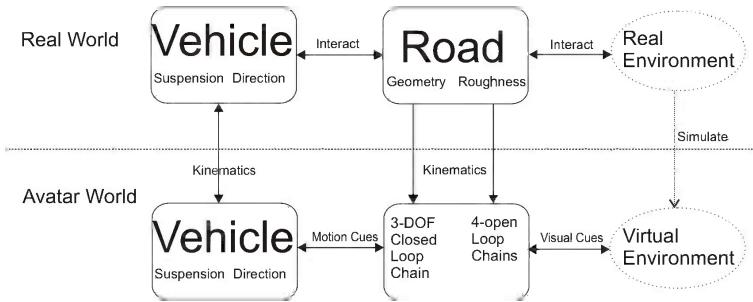


Fig. 1 The correlation between the physical entities and avatar entities that define CAV-Drive

According to the relationships among entities described in fig. 1 the concept of CAV-Drive is based on describing abstracted entities with respect to the physical ones using mathematical models through which the motion and visual cues can be simulated. Thus, the corresponding avatar entities are expressed as:

- the vehicle is described by four simulated suspension models and direction
- the runway: the road geometrical parameters impose a 3-DOF closed loop chain (mechanism) for vehicle direction handling, while the vertical excitations that influence the vehicle suspension due to road roughness impose 4-open loop chains

1.2 State of the Art

Driving simulators are complex systems used for simulating and reproducing the effect of driving in a controlled environment. Developing driving simulators is a demanding task and requires a large amount of investment in time for structural implementation and resources as the costs of equipment can be overwhelming [5], especially when dealing with high-fidelity simulators. Usually, validating driving simulators refers to their hardware and mechanical structure, but one of the most demanding challenges is to simulate the vehicle dynamic model as well as retaining its mechanical properties in order to obtain realistic results. There are many phenomena with non-linear behaviour that influence the sensations that appear when driving (the contact between the tire and the road surface, for example) and because of that, are difficult to be computed using an accurate mathematical model. Simulated vehicle models are extensively studied in literature, but most virtual models

are non-deterministic, while the real models are not [10]. Even so, the concept of a fully simulated driving simulator, described as an avatar, can be useful for a user in having a realistic feedback and a interactive window to test new concepts [11] in the process of virtual prototyping for full-scale driving simulator. The current approaches deal with validations for road design/roughness or simulated vehicle models, in which the suspensions are usually represented using a classical spring-mass damper. There are several types of dependent and independent suspensions and also steering mechanisms currently used in the car manufacturing [1], [12]. Using kinematic analysis, several virtual suspensions were implemented by [9] and [8] with promising results. Furthermore, different types of runway models with respect to road roughness where described in [7], [13], while the road geometry was classified and modelled in a modular manner and can be used for simulations [17]. The mechanical structures that offers insight to reproducing vehicle motions are numerous and were heavily discussed and described in [3]. Considering the entities presented in 1.1 which describe the complete road-vehicle model, results from simulated models from each corresponding component were presented, this is because a complete road-vehicle model is yet to be developed.

Each entity will be presented in the following chapters focusing on the set of parameters required to be implemented. Chapter 2 deals with the description of a road profile and how this profile can be used as an input for generating motion cues. Next, in chapter 3 a 3-DOF parallel mechanism (closed chain) is presented which will be used as motionbase. In chapter 4 a full road-vehicle model with suspensions is presented along with a suspension mechanism, whilst in chapter 5 the CAV-Drive concept is developed by integrating all the entities and validated through numerical results. In chapter 6 briefly conclude the CAV-Drive concept.

2 Mathematical Road Modelling

The road or the runway is considered one of the most important excitation factors for a road simulator, this is because while driving disturbances may occur from the driver's steering actions and at the level of the tyre/suspension systems (road irregularities - roughness, holes and others) [4]. These disturbances are described by vibrations, forces and torques. In automotive industry, they are very important mainly in driver safety and are felt by the driver from excitation sources such as the road profile. Thus, in an simulated environment accurate representation of different types of roads leads to obtaining realistic values for the forces that act upon the wheel. The influence of the road profile was discussed in [14]. Furthermore, the comfort of driving a vehicle is related to the effects of the road profile. These effects are produced due to several parameters whose values are categorized by ISO [15]. The road surface refers to the existence of profiles that describe road irregularities. Road roughness is the main parameter that characterize different categories of standardised roads and the accurate computation of this parameter contributes significantly in increasing the realism of a simulator [6]. Some mathematical aspects with respect to road roughness are presented. There are several mathematical

relations of road roughness [13]. Usually road roughness is expressed using the Power Spectral Density ψ described by [7]:

$$\psi(\Omega) = \begin{cases} a/(2\pi\Omega)^2 & \Omega \leq 1/2\pi \\ a/(2\pi\Omega)^{1.5} & \Omega \geq 1/2\pi \end{cases} \quad (1)$$

The spatial angular frequency Ω values are taken into consideration according to the road category, as detailed in [15]. Using the Power Spectral Density ψ , the discrete form for roughness is given as:

$$z'_r(t) + \omega_0 * z_r(t) = \sqrt{\psi(\Omega_0)uw(t)} \quad (2)$$

where:

1. z_r is the road roughness
2. ω_0 is the reference angular frequency given in rad/s
3. Ω_0 is the reference spatial angular frequency given in rad/m
4. u represents the constant speed of a vehicle
5. $w(t)$ white noise signal with power spectral density being 1

It is worth mentioning that not only the road roughness has an effect on the vehicle, but also the speed u at which this vehicle is moving on such a profile can have an important impact that leads to an increase/decrease of vibrations. The range of values for the above parameters categorize the road roughness into 5 types of roads ranging from A-category roads (highways) to E-category roads (country roads). An A-category road profile is presented in fig. 2.

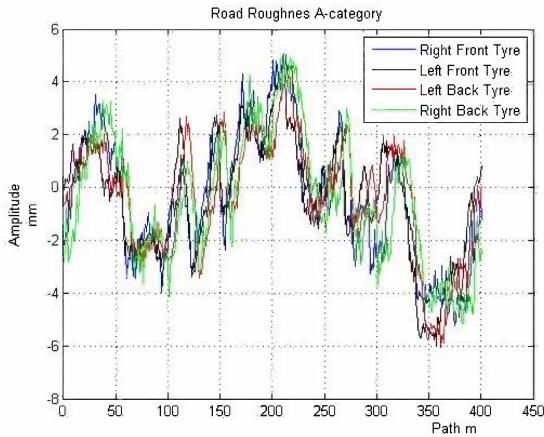


Fig. 2 A-category Road Roughness

3 The Motionbase Platform

When a vehicle is moving along a runway several movements among its relative components appear. This can have a major effect when dealing with putting a full-vehicle model on a motionbase. The vehicle model with forces and torques that appear when moving at tire level is represented in fig. 3. It may be emphasized that the forces F_x , F_y and the torque M_z at the vehicle's center of gravity can be simulated using a corresponding mechanism. Such mechanism is formed by a planar mechanism (with 3-DOF) and 4 serial loops, each of them with 2-DOFs. Again, the generalized forces (F_i , M_i) can be developed by the four serial loops, each of them being inputs for the wheel. An important remark is that the generalized forces can reproduce both road geometric roughness and tire-road contact forces.

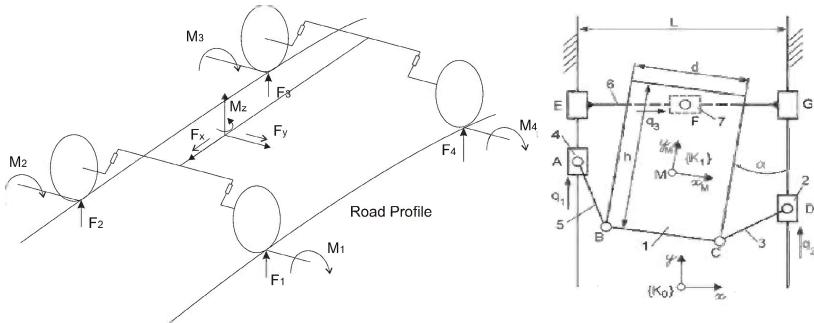


Fig. 3 Forces that act on a vehicle and the corresponding 3-DOF mechanism

The mechanism described in fig.3 is a planar parallel mechanism. The kinematic analysis of such a mechanism considers a base coordinate system K_0 which is attached to the fixed frame at the midpoint of the length L . The coordinate system - K_1 , is attached to the mobile platform of the mechanism, at the point M . The direct kinematic problem can be solved using the cut body method [16]. The input kinematics variables (known variables) for the direct pose problem are:

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad (3)$$

where q_1, q_2 and q_3 describe the movements of the sliders A , D and F , respectively. The coordinates of the end-effector located at point M are described as functions of q_1, q_2 and q_3 :

$$\begin{aligned} x_M &= f_1(q_1, q_2, q_3) \\ y_M &= f_2(q_1, q_2, q_3) \\ \alpha &= f_3(q_1, q_2, q_3) \end{aligned} \quad (4)$$

The input kinematics variables for the inverse pose problem are the coordinates of the end-effector M (x_M, y_M, α). In this case the displacement values for q_1, q_2 and q_3 are computed using the current geometrical parameters, see fig. 3. The solutions are:

$$q = \begin{cases} (y_M - ((h/2)\cos(\alpha) + (d/2)\sin(\alpha)) + \sqrt{l_1^2 - x_M + L/2 - a^2}) \\ (y_M - ((h/2)\cos(\alpha) - (d/2)\sin(\alpha)) + \sqrt{l_1^2 - (L/2 - x_M - b)^2}) \\ (x_M + L/2) + m \sin(\alpha) \end{cases} \quad (5)$$

4 Simulated Suspension Model

Suspension systems allow relative motions from the wheel to the body chassis of a vehicle. These relative motions are discussed in [1]. There are many suspension mechanisms, each of them contribute to the vehicle maneuverability and driver safety and comfort. A suspension model is presented in fig. 4 which is a variant of the MacPherson suspension [2], but other types can be implemented. The full-vehicle model has four suspensions models. Each suspension produces an independent response from each other given a road profile excitation.

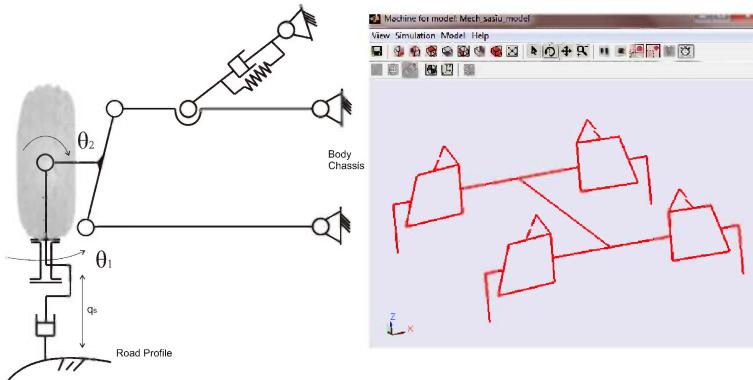


Fig. 4 Suspension Model: one suspension model and vehicle suspension model

The suspension mechanism in fig. 4 is described by the following parameters:

- q_s the road response input which lets the suspensions to have vertical motions
- θ_1 - considered to be the steer angle of the tire
- θ_2 - the camber angle is the angle of the wheel relative to vertical axis of the road [1]

5 Numerical Results

In order to validate the CAV-Drive concept, it was proposed to simulate the motions for the vehicle when moving in a straight line with the runway being described as an A-category road with the profile from fig. 2. Thus, it is considered:

- for the runway, the laws of motion are imposed for only for q_1, q_2 , with $q_1=q_2$, $q_3 = 0$ and $q_1 = f(t)$ with $f = at + b$, $t = [0 \dots 10]$, see fig.3
- four independent vehicle suspensions based on fig.4 with the following suspension parameters:
 - Spring Stiffness of suspension, $k = 200$ N/m
 - Damping constant of suspension, $b = 30$ Ns/m

The block diagram for CAV-Drive was implemented using Matlab Simulink (fig.5) with the avatar presented in fig.6.

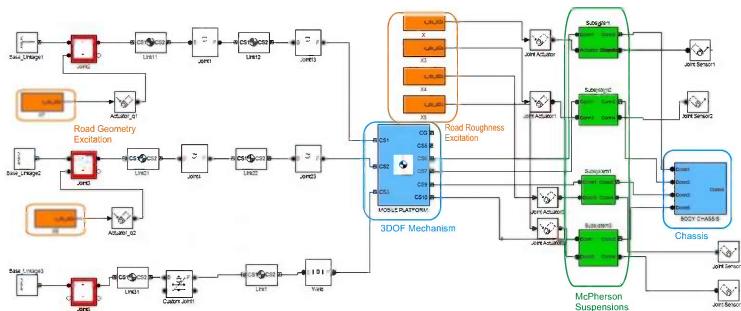


Fig. 5 Simulink implementation of CAV-Drive concept

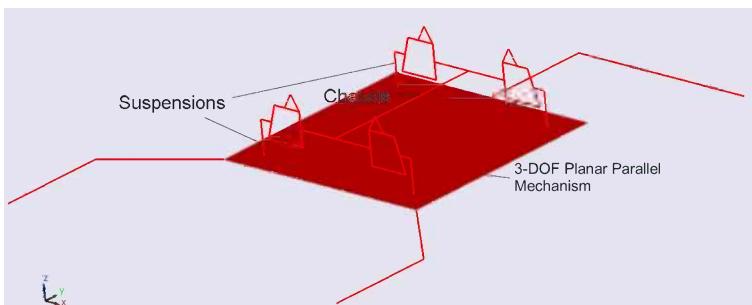


Fig. 6 Simulink display of CAV-Drive concept

Based on these considerations the aim is to determine the linear and angular acceleration that act on the chassis' center of gravity. These accelerations are presented in figs. 7 and 8.

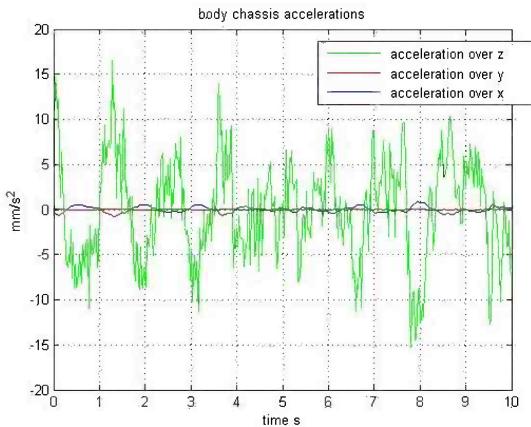


Fig. 7 Values for linear accelerations that act on the chassis

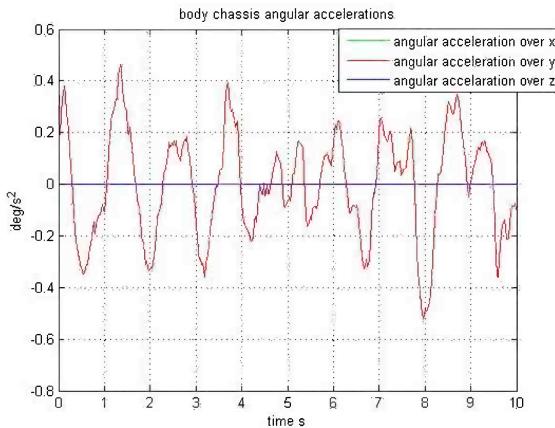


Fig. 8 Values of angular accelerations that act on the chassis

The values for the accelerations are obtained according with the values determined and described in specialized literature [7].

6 Conclusions

The paper proposes and describes the concept of CAV-Drive as a fully simulated model for driving simulators by integrating a computed road profile, four open loop chains that respond to the road excitations, a parallel planar manipulator which moves the vehicle and a full-vehicle model with suspensions that simulates the

dynamic behaviour. This avatar model can be a very powerful assistance tool that can be designed to ensure feedback in an interactive manner, this is because the avatar allows the implementation of different types of suspension models by changing the suspension parameters for each wheel and can simulate different types of road profiles (geometry and roughness) without having to physically develop them.

Acknowledgements. Research supported by the grant 149/2011.

References

1. Jazar, N.R.: *Vehicle Dynamics: Theory and Applications*. Springer, New York (2008)
2. Rill, G.: *Vehicle Dynamics. Lecture Notes*, Regensburg, Germany (2008)
3. Merlet, J.-P.: *Parallel Robots*. Springer, The Netherlands (2006)
4. Bogsjo, K., Podgorski, K., Rychli, I.: Models for road surface roughness. *Vehicle System Dynamics: International Journal of Vehicle Mechanics and Mobility* 50(5), 725–747 (2012)
5. Weinberg, G., Harsham, B.: Developing a low-cost driving simulator for the evaluation of in-vehicle technologies. In: *Proceedings of the 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pp. 51–54 (2009)
6. Pacurari, I.-R.: Cercetari teoretice si experimentale privind dezvoltarea simulatoarelor de drum (romanian). *Theoretical and Experimental Research for road simulators*. PhD Thesis, Universitatea Tehnica din Cluj-Napoca, Cluj-Napoca, Romania (2011)
7. Gonzalez, A., O'Brien, E.-J., Lia, Y., Casheel, K.: The use of vehicle acceleration measurements to estimate road roughness. *J. Vehicle System Dynamics* 46(6), 483–499 (2008)
8. Mántaras, D., Luque, P., Vera, C.: Development and validation of a three-dimensional kinematic model for the McPherson steering and suspension mechanisms. *J. Mechanism and Machine Theory* 39(6), 603–619 (2004)
9. Papegay, Y., Merlet, J.-P., Daney, D.: Exact kinematics analysis of Car's suspension mechanisms using symbolic computation and interval analysis. *J. Mechanism and Machine Theory* 40(6), 395–413 (2005)
10. Schwarz, C.: *Validating Vehicle Models*. In: *Handbook of Driving Simulation for Engineering, Medicine, and Psychology* (2011)
11. Kohler, T., Matzler, K., Fuller, J.: Avatar-based innovation: Using virtual worlds for real-world innovation. *J. Technovation* 29, 395–407 (2009)
12. Alexandru, P., Visa, I., Talaba, D., Alexandru, C.: Antonya, C.: Modelarea Statico-Dinamica a Mecanismelor de Ghidare ale Rotilor automobilelor (romanian). In: *Static and Dynamic Modeling of the Guiding Mechanisms for Automobile Wheels*, Lux Libris, Brasov, Romania (2005)
13. Schiehlen, W.: White noise excitation of road vehicle structures. *J. Sadhana* 31(4), 487–503 (2006)
14. McManus, K.J., Mann, A., Evans, R.P.: The Analysis of Road Roughness and the Perception of Road Roughness. In: *5th International Symposium on Heavy Vehicle Weights and Dimensions*, pp. 1–11 (1998)
15. ISO 8608: Mechanical vibration-Road surface profiles-Reporting of measured data (1995)

16. Brisan, C.: Aspects of reconfigurability of a special class of parallel robots. In: 2007 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp. 1–6 (2007)
17. Brisan, C., Vasiu, R.-V., Munteanu, L.: A modular road auto-generating algorithm for developing the road models for driving simulators. *J. Transportation Research Part C: Emerging Technologies* 32, 269–284 (2013)

Interconnection of Federated Clouds

Massimo Ficco, Luca Tasquier, and Beniamino Di Martino

Abstract. Cloud Federation is an emerging computing model where multiple resources from independent Cloud providers are leveraged to create large-scale distributed virtual computing clusters, operating as within a single Cloud organization. This model enables the implementation of environmental diversity for Cloud applications, and overcomes the provisioning and scalability limits of a single Cloud, by introducing minimal additional cost for the Cloud consumer. In such a scenario, it is necessary to leverage specific middleware technologies that enable the effective support of inter-Cloud communication services between Cloud providers. This paper proposes an interconnection solution for Cloud Federations based on publish-subscribe services.

1 Introduction

In the last years, the Cloud services market experienced an extremely rapid growth, as reported in several market research reports, which may lead to severe scalability problems [12]. Therefore, in order to cope with the resource capacity limits of a single Cloud provider, as well as to address the vendor lock-in problem associated to the choice of a single proprietary Cloud solution, the concept of federating multiple heterogeneous organizations is receiving an increasing attention by the key players in the Cloud services market [2]. Cloud Federation extends the scalability of Cloud systems by introducing extreme elasticity in resource management, and hence, enabling service requests to be satisfied also in presence of rapid increasing consumer demand and heavy usage of the infrastructure.

On the other hand, the main problem in Cloud Federation is to seamlessly and transparently join, from the operational point of view, two or more Clouds,

Massimo Ficco · Luca Tasquier · Beniamino Di Martino
Department of Industrial and Information Engineering,
Second University of Naples, via Roma 29, 81031 Aversa, Italy
e-mail: {massimo.ficco, luca.tasquier,
beniamino.dimartino}@unina2.it

characterized by substantial differences in their organizations. Therefore, a large effort is being spent within the Cloud Computing community to identify open solutions and standards for Clouds interoperability, such as the Open Cloud Manifesto [17], Open Cloud Computing Interface (OCCI) [6], Open Cloud Standards Incubator [16], and the Cloud Data Management Interface (CDMI) [19]. However, although significant benefits may result from the interconnection of multiple Clouds into an uniform way, the open and dynamic nature of these systems coupled with their heterogeneity, makes the communication in inter-Cloud environments an extremely challenging task. Thus, a suitable network virtualization framework is needed to deal with the heterogeneity of the different available Cloud solutions in lack of an uniform data communication model. Specifically, in order to enable what is called in the literature “*Cloud Federation*”, *i.e.*, empowering the run of distributed applications on resources located across independent Cloud infrastructures, it is essential for all the resources belonging to the cooperating Clouds to be able to communicate with each other, by acting as virtual nodes operating within a single distributed organization. Therefore, in this paper we envision a federation solution based on the well-known *publish-subscribe service* [8]. We propose an inter-Cloud middleware, supporting the seamless interconnection of distributed resources in different administrative domains.

2 Related Work

Cloud Federation is a research topic of great interest. At the state of art, many research works and projects have been devoted to Cloud Computing issues related with the concept of federation. Several vendor-agnostic open-source solutions have been proposed, like LibCloud [21], DeltaCloud [22], jClouds [23], SimpleCloud [15], mOSAIC [5, 11], and MODAClouds [1]. However, they mainly present approaches, abstractions and toolkits for the design and execution of applications on multiple Clouds that aim at supporting system developers and operators in exploiting multiple Clouds and in migrating their applications from Cloud to Cloud as needed. For example, mOSAIC provides a Software Platform and a Cloud Agency for supporting the Cloud developers during the development and deployment of a Cloud application in a Sky computing infrastructure. In particular, it introduces supplementary layers of abstractions that offer uniform access to heterogeneous Cloud resources, independently from the Cloud provider and the technologies it supports, allowing applications to work with abstract Cloud resources. However, only simple pipes and shared memory are provided to support peer to peer communication at application level between distributed components. The LibCloud project is a client library for accessing Clouds, supporting a wide range of providers, implemented in Python and Java. Analogously, jClouds, is another stable library implementation targeting portable abstractions for Cloud access. DeltaCloud is an open source project developing an API to deal with several Cloud service providers. It targets REST access and backward compatibility across versions, providing long-term stability for

scripts, tools and applications. Being a service-based REST API, it works like an intermediary layer, receiving commands and sending them through specially created Cloud drivers, which provide direct operation mapping with the provider's API.

To the best of our knowledge, ViNe [24] is the only work that addresses connectivity among virtual components distributed across several Cloud administrative domains. ViNe incorporates the necessary mechanisms for recovering full connectivity among ViNe-connected nodes, which could potentially be distributed across several Cloud administrative domains. However, it has not been conceived as a standard solution, but only as an embedded approach for deployment of user-level virtual routers for implementing overlay network communication.

3 Publish-Subscribe Service for Clouds Interconnection

The proposed publish-subscribe service-based solution, enables the interconnection of multiple Cloud organizations scattered throughout the network. It provides an integration layer that manages the heterogeneity among the Clouds without the necessity of implementing several interfaces and adapters requiring heavy coding efforts and complex software artifacts. The proposed middleware represents an efficient solution for Cloud interoperability thanks to the decoupling and flexibility offered by the publish-subscribe interaction model, that can be effectively exploited in order to deal with Cloud orchestration and the definition of advanced aggregated services on top of the ones offered by the single federated Cloud organization.

The adopted publish-subscribe service offers data-centric communications based on an event-based messaging model. The clients can play two distinct roles: *publishers* that produce information (*e.g.*, available data sets, runtime resource, storage space) and distribute the associated notifications, and *subscribers* that consume notifications in which they are interested. A notification is the act of delivering an event. The interest of subscribers in the published notifications are specified in terms of subscriptions (consisting of several predicates). Typically however, a client can take on both publisher and subscriber roles. The *brokers* are the entities, located on the different sites belonging to the federation, that perform matching and forwarding of messages with respect to a certain event schema (*e.g.*, the advertisement tree or message type) and according to specific error recovery policies. They realize a virtualized network infrastructure overlaid on top of the physical connections, which supports the communication between the publisher and subscriber entities operating within the federation. The main strength of publish-subscribe systems is their ability of ensuring asynchronous communication among the involved nodes that interact in a loosely decoupled way, resulting in an extremely scalable infrastructure for information exchange and distributed workflows.

The proposed integration layer articulates according to a well known schema based on two fundamental building blocks that can be implemented through general reusable software solutions or design patterns:

- the *Adapter* resolves the technological heterogeneity of each Cloud with respect to the integration layer according to a “*lingua-franca*” approach, *i.e.*, a common

language whose format and syntax is used for the communications between the Clouds. Each Cloud has an adapter from the common language to its own one.

- the *Broker* is a mediation component between the Clouds, containing the whole integration layer logic. In this manner, it is not necessary to explicitly connect each Cloud with the other ones by an ad-hoc connector, but only plug its Adapter to the Broker, without bringing any change to the integration solutions and/or the Broker itself. Therefore, the integration is transparent from the perspectives of both the individual integrated Clouds and the resulting federated solution.

Figure 1 shows the proposed solution, where the front-end of each Cloud is interconnected to an Adapter, and the Adapters are interconnected with the Brokers through a publish/subscribe service (in order to improve the offered scalability and reliability, the federation can encompass more than one Broker). The integration logic within the Broker and the Adapter is structured by means of well known Enterprise Integration Patterns (EIP) [13], resulting into an efficient implementation. In particular, the EIPs are implemented by using a versatile modular integration framework, such the Apache Camel [14]. Such framework allows us to define routing and mediation rules in a variety of domain-specific languages. Moreover, in our realization of the integration layer, we extended this solution by considering a component-oriented approach for the development of the Broker. Therefore, we do not limited ourselves in using only an integration framework, but relied an Enterprise Service Bus (ESB) [18] for designing and implementing the interaction and communication between mutually interacting software components in service-oriented architecture (SOA). Such a solution is typically already equipped with an integration framework, but in addition it provides a complete component container for realizing the integration through component-oriented programming. This facilitates the realization of the Broker, since we can use the available implementation of the components we may need, as well as it allows to include the Adapters already available in the IT market for most of the adopted technologies, such as Web Services, CORBA or RMI. The result is a considerable limitation of the amount of code to make, and the adoption of standard and open-source frameworks. Moreover, it increases the flexibility in the number and kind of information sources to be integrated.

In addition, we have considered an asynchronous communication pattern between the Brokers and all the integrated components (realized as an ESB). This allows us to have a scalable and easy solution to dynamically plug new information sources. An event-driven middleware [8] based on a well-known standard, such as Java Service Message (JMS) [20] has been used for this purpose. It provides content-based filtering by means of selector strings expressed with a subset of the SQL92 conditional expression syntax. Therefore, with a publish-subscribe service the Clouds can articulate requests toward the federated components by using notifications. Moreover, XML is the adopted lingua-franca within the considered Cloud federation and the format used for the notifications exchanged among the Adapters by the adopted publish-subscribe service [4]. Therefore, the adapter implementation can be assumed in terms of EIP:

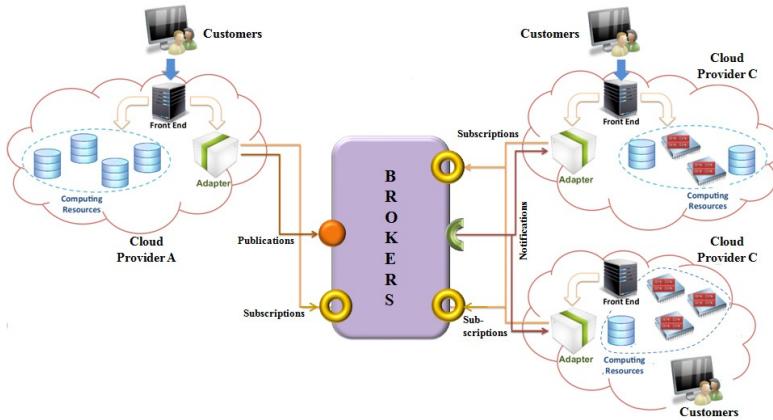


Fig. 1 Cloud federation architecture based on a publish/subscribe service

- An incoming Message Endpoint is realized in order to accept messages incoming from the front-end of a given Cloud. There is also a similar, but outgoing, Message Endpoint for sending messages toward the interior front-end of the federated Cloud. Such endpoints are coded based on how the front-end has been realized, *e.g.*, through a socket or a CORBA object.
- A Message Translator performs the conversion of the received data in the internal format used by the federated Cloud, to a notification in XML and of performing the dual operation for passing from XML to the internal format.
- Two components are present so as to publish the obtained notifications and/or to receive notifications published by the other Clouds within the federation.

In such a federation, there are several general key requirements that the adopted publish/subscribe service-based solution should satisfy, including scalability, reliability, performability, and security [7].

4 Conclusions

This paper presents a flexible federated Cloud architecture, in which the cooperation between the involved organizations is based on a scalable publish-subscribe middleware for dynamic and transparent interconnection of multiple types of resources and entities. In the future work we aim at designing specific techniques for monitoring performance and security aspects [10, 9, 3].

Acknowledgements. This work has been supported by mOSAIC (grant FP7-ICT-2009-5-256910), Collaborating Smart Solar-powered Micro-grids (CoSSMic - grant FP7-608806) projects, and PRIST 2009, “Fruizione assistita e context aware di siti archelogici complessi mediante terminali mobile”, founded by Second University of Naples.

References

1. Ardagna, D., et al.: Modaclouds: A model-driven approach for the design and execution of applications on multiple clouds. In: 2012 ICSE Workshop on Modeling in Software Engineering (MISE), pp. 50–56 (2012)
2. Buyya, R., Ranjan, R., Calheiros, R.: InterCloud: Scaling of Applications across multiple Cloud Computing Environments. In: Proceedings of the 10th Int. Conf. on Algorithms and Architectures for Parallel Processing (2010)
3. Casola, V., Cuomo, A., Rak, M., Villano, U.: Security and performance trade-off in perfcloud. In: Guerraccino, M.R., et al. (eds.) Euro-Par-Workshop 2010. LNCS, vol. 6586, pp. 633–640. Springer, Heidelberg (2011)
4. Cilardo, A., Coppolino, L., Campanile, F., Romano, L.: Adaptable parsing of real-time data streams. In: Conf. on Parallel, Distributed and Network-based Processing, pp. 412–418 (2007)
5. Di Martino, B., Petcu, D., Cossu, R., Goncalves, P., Máhr, T., Loichate, M.: Building a mosaic of clouds. In: Guerraccino, M.R., et al. (eds.) Euro-Par-Workshop 2010. LNCS, vol. 6586, pp. 571–578. Springer, Heidelberg (2011)
6. Edmonds, A., Johnston, S., Metsch, T., Mazzaferro, G.: Open Cloud Computing Interface - Core & Models (2010), <http://occi-wg.org/about/specification/>
7. Esposito, C., Ficco, M., Palmieri, F., Castiglione, A.: Interconnecting Federated Clouds by Using Publish-Subscribe Service. Cluster Computing, 1–17 (2013)
8. Eugster, P., Felber, P., Guerraoui, R., Kermarrec, A.M.: The many Faces of Publish/subscribe. ACM Computing Surveys 35(2) (2003)
9. Ficco, M.: Security event correlation approach for cloud computing. Journal of High Performance Computing and Networking 7(3) (2013)
10. Ficco, M., Romano, L.: A generic intrusion detection and diagnoser system based on complex event processing. In: Proceedings - 1st Int. Conf. on Data Compression, Communication, and Processing, pp. 275–284 (2011)
11. Ficco, M., Venticinque, S., Di Martino, B.: mOSAIC-based intrusion detection framework for cloud computing. In: Meersman, R., et al. (eds.) OTM 2012, Part II. LNCS, vol. 7566, pp. 628–644. Springer, Heidelberg (2012)
12. Gartner: Forecast: Platform as a Service, Worldwide, 2010–2015, 3Q11 Update (2011), <http://www.gartner.com/id=1792219>
13. Hohpe, G., Woolf, B.: Enterprise Integration Patterns. Addison-Wesley Professional (2003)
14. Ibsen, C., Anstey, J.: Camel in Action. Manning Publications (2011)
15. Inc, Z.T.: Simple Cloud API (2012), <http://simplecloud.org/>
16. Incubator, O.C.S.: Cloud Management Initiative (2011), <http://www.dmtf.org/standards/Cloud>
17. Open Cloud Manifesto Community: Open Cloud Manifesto (2009), <http://www.opencloudmanifesto.org>
18. Rademakers, T., Dirksen, J.: Open-Source ESBs in Action: Example Implementations in Mule and ServiceMix. Manning Publications (2008)
19. SNIA: Cloud Data Management Interface (2012), <http://www.snia.org/cdmi>
20. Snyder, B., Bosanac, D., Davies, R.: ActiveMQ in Action. Manning Publications (2011)
21. The Apache Software Foundation: Apache Libcloud Python library (2011), <http://incubator.apache.org/libcloud>
22. The Apache Software Foundation: Deltacloud API (2011), <http://deltacloud.apache.org/>
23. The Apache Software Foundation: jClouds (2011), <http://code.google.com/p/jclouds>
24. Tsugawa, M., Matsunaga, A., Fortes, J.: User-level virtual network support for sky computing. In: Fifth IEEE Int. Conf. on e-Science, e-Science 2009, pp. 72–79 (2009)

Effective QoS Monitoring in Large Scale Social Networks

Luigi Coppolino, Salvatore D'Antonio, Luigi Romano,
Fotis Aisopos, and Konstantinos Tserpes

Abstract. Social Networking activities are still occupying the majority of the time that Internet users are spending in the Web. The generated content and social dynamics represent precious resources that everybody wishes to control. This scenario poses several challenges including the fact that different implementations, technologies, and formats are used to manage web content and social dynamics in heterogeneous, often antagonistic, Social Networking Sites. In order to master this heterogeneity the SocIoS project has defined an API that enables the aggregation of data and functionality made available by different Social Networking Sites APIs and their combination into complex and novel application workflows. However, the dependency on Social Networking Sites does not allow users of the SocIoS API to control the Quality of Service provided by the underlying platforms. In this paper we show how the QoSMONaaS (QoSMONitoring as a Service) component can be used to monitor and evaluate relevant metrics, such as availability and response time of the API calls, that are specified in the Service Level Agreement document. QoSMONaaS has been developed within the context of the SRT-15 project to implement a dependable (i.e. unbiased, reliable, and timely) monitoring of Quality of Service.

Keywords: SLA Monitoring, Social Networks, QoS Monitoring.

Luigi Coppolino · Salvatore D'Antonio · Luigi Romano
University of Naples Parthenope, Naples, Italy
email: {luigi.coppolino, salvatore.dantonio,
luigi.romano}@uniparthenope.it

Fotis Aisopos · Konstantinos Tserpes
National Technical University of Athens, Athens, Greece
email: {fotais, tserpes}@mail.ntua.gr

Konstantinos Tserpes
Harokopio University of Athens, Athens, Greece
email: tserpes@hua.gr

1 Introduction

Social Networks (SN) are the ideal future service marketplaces. SN users are increasing at a tremendous pace with Web 2.0 and SN Sites (SNS) have attracted more than 500 million regular users within just their first 5 years of existence. The number of the potential customers is huge, coming from almost every societal class, cultural background, and age. The requirements are modest, namely a computer, a browser, network access, and the natural need for socializing. Taking advantage of the social dynamics as well as the vast volumes of amateur content generated every second in a SNS, is a fundamental step towards creating a potentially huge market of services. Providing developers with cross-platform tools that enable them to manage the dynamically generated content and complex social interactions will create an agile and profitable market of services and will bring the Internet of Services concept a step closer to realization. Such enabling tools will make it possible to build, deploy, and potentially sell services that combine data and functionalities from two or more different SN services disregarding the underlying SN implementation. Such challenging issues are currently addressed by the EU FP7 SocIoS project [1] that aims at paving the way for building qualitative, functional and usable business applications exploiting the User Created Content and the Social Graph of users in Social Networks. The main artifact that SocIoS presents is a framework (such as an API) for allowing application developers of variable level of expertise to combine content and services from a wide range of SN sites into complex workflows. The approach for services development and deployment is “write once, run everywhere”. The resulting services can be deployed on the developer’s own server or inside the SocIoS framework. There they can be discovered and used by other users of the platform, allowing SocIoS to orthogonally address the debate regarding interoperability through a Trusted Third Party (TTP) or an ad-hoc, distributed solution. The terms of service usage, including billing information, will be defined by Service Level Agreements (SLAs). It is thus necessary to guarantee that breaches of such SLAs are immediately detected. If the root-cause of the breach is accurately detected, then the proper reaction strategy can be implemented and the liability of each party involved in the transaction can be identified. Instead of developing its own QoS monitoring solution, the SocIoS framework makes use of the QoS monitoring component developed within another FP7 project, namely the SRT-15 project [2]. QoSMONaaS (QoSMONitoring as a Service) [3,4,5,6] is an implementation of a dependable (i.e. unbiased, reliable, and timely) business process level monitoring framework.

In this paper we present the integration of the two frameworks, and describe how QoS-MONaaS can be used to monitor a typical SocIoS application. The paper is structured as follows. Section 2 presents the SocIoS framework. Section 3 describes the conceptual architecture of QoSMONaaS. The integration between the SocIoS framework and QoSMONaaS is presented in Section 4, while Section 5 provides some concluding remarks and indications about future work.

2 The SocIoS Framework

The SocIoS framework is a software stack that operates on top of SNSs with the purpose of: (i) aggregating data and functionality from a multitude of underlying social media platforms, (ii) providing a tool for developers to build social analytics services on top of the supporting social media platforms, and (iii) accommodating newly created applications that use the abovementioned services and provide them as through usable interfaces.

With the proper configuration and development of intermediate services, the framework can support any application that requires the harvesting of social media, filtering the content with sophisticated features while at the same time harnessing the scale issues of the endeavour (volume of data, number of users and platforms).

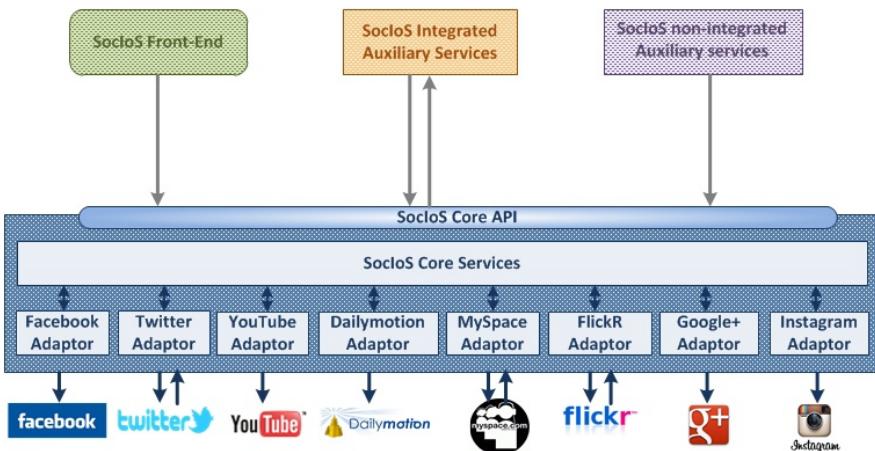


Fig. 1 SocIoS Framework High Level Architecture

The objectives mentioned above are achieved through a layered Service-Oriented Architecture (SOA) which is depicted in Fig. 1. The SocIoS framework consists of several main entities, such as the SocIoS Object Model, the Core Services, the Auxiliary Services and the Front-End. A short description of these components is given below.

The **SocIoS Object Model** (SOM) [7] is an aggregation of methods provided by underlying SNS APIs. The **SocIoS API** maps a standard interface to collections of methods and objects of the social networking site APIs. The SOM defines both a meta-API (SocIoS API) and a data model.

The **Core Services** are a set of adaptors that implement the SocIoS API methods as well as the data transformation between the supported SNSs' APIs and the higher service layers (SocIoS Auxiliary Services).

The **Auxiliary Services** are third party services that extend the functionality of the core services and use the same data models as the core services. Depending on their purpose, their operations may or may not be exposed as part of the SocIoS API in the sense that they can extend it. This differentiates them from integrated and non-integrated Auxiliary Services. In general, they provide analytics to the data delivered by the Core Services and even though they can be re-used, they were developed with the purpose to meet the requirements of a certain application. Some examples are:

- Media Item Ranking and Recommendation (Integrated): A service for assessing the subjective value of a media item to a specific user,
- Topic-Specific Community Detection (Integrated): A service for identifying social media user communities that are implicitly linked to each other,
- Event Detection (Integrated): A service for highlighting intense or unusual activity within a community,
- Social Filtering (Integrated): A service for managing social media user groups,
- Translation (Integrated): A service for translating textual content in various languages,
- The FlexiPrice service (non-Integrated): A service that enables two users (a buyer and a seller) to set a price of a content item,
- The Crowdsourcing Game (non-Integrated): A service that allows the posting of crowdsourcing tasks in the form of a game.

Finally, the **SocIoS Front-End** provides a user interface for interacting with the components as well as an authentication mechanism. It also deals with the user rights and privacy settings while in certain applications, it may also serve as a point of integration for the Core and Auxiliary Services.

3 QoSMONaaS Conceptual Architecture

QoSMONaaS was initially thought as an application running on top of the cloud platform developed within the FP7 SRT-15 project (Stream Routing Technology for 2015) [2]. The main objective of the SRT-15 project is to build a scalable platform for connecting Future Internet (FI) business applications and services. The platform will enable the discovery and integration of dynamic enterprise services on the Internet. Furthermore it will allow for dependable and scalable cloud-based processing of data coming to and from a variety of heterogeneous enterprise services spread across multiple distributed locations. In order to be able to embrace the change in the enterprise information processing landscape, the SRT-15

platform relies on technologies that support rapid change, i.e. cloud computing, content-based routing [8] and complex event processing [9]. Within the SRT-15 platform, QoSMONaaS implements a dependable QoS monitoring facility, which is made available as a service to third parties willing to access monitoring services. QoSMONaaS was designed to be easily ported on top of other cloud platforms. Portability was achieved by relying on a clean cut approach, which clearly identifies: (i) the interface that QoSMONaaS shares with other applications (referred to as the "Basic Interface"), that is the interface that all applications use to request platform services; (ii) the interface that QoSMONaaS uses to gather information which is specifically needed for the purpose of QoS monitoring. This information is not provided to other applications. Thus, this interface is referred to as the "Extended Interface"; (iii) two key services, namely Authentication and Anonymization, that QoSMONaaS uses from the underlying platform, and which might need some modifications and/or require some development efforts if QoSMONaaS is ported on top of a different cloud platform (this is referred to as the "Platform-Adaptation layer").

More details on QoSMONaaS architecture can be found in [3]. In order that QoSMONaaS is able to monitor the actual QoS level delivered by the Service Provider to the Service User, the Service Provider has to provide the underlying cloud platform (or the Platform-Adaptation layer) with both a formal description of the Service Level Agreement, containing information about the Key Performance Indicators (KPIs) which are of interest, and a formal description of the specific business process. The Service User will then request the monitoring service, in the same way as it would request any other service. As already mentioned, QoSMONaaS uses the service provided by an anonymization system in order to avoid that the real identity of monitored parties be revealed to the monitoring application. This makes the QoSMONaaS application "trusted by design" since it would not be able to act in favor of one of the parties. The basic idea is that since the monitoring tool ignores the real identities of the involved parties, it cannot cheat. In the current implementation we use a simple scheme, which is in all respects similar to those used in many social network applications [10]. Basically, the approach relies on the simple graph-anonymization technique, which replaces the identifying information of the parties with random identifiers.

4 SocIoS – QoSMONaaS Integration

The SocIoS framework (Fig. 2) allows users to get data from SNs according to a predesigned workflow.

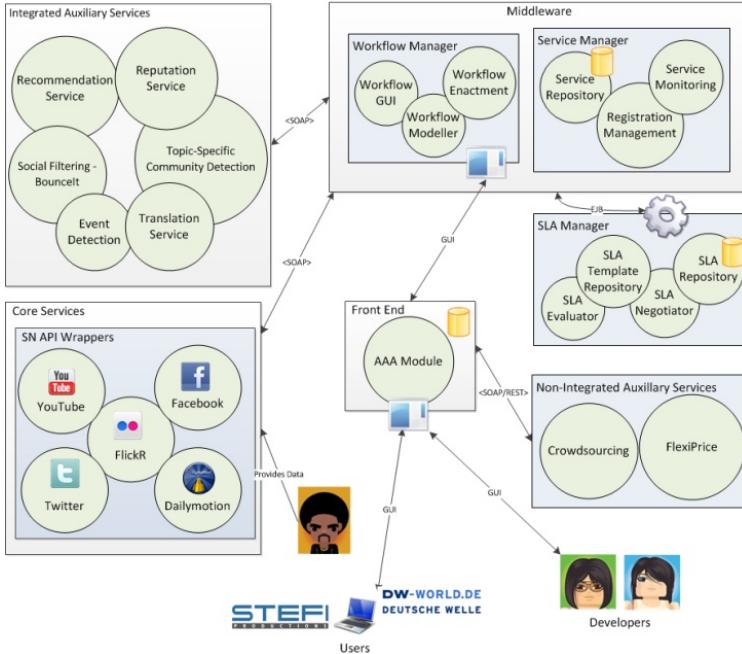


Fig. 2 SocIoS framework logical view

The workflow describes: i) the data sources, and ii) what kind of auxiliary services has to be applied to filter the gathered data before some of them reach the end user. Workflow execution can be regulated by some SLAs describing the QoS to be guaranteed to the end user. As an example, some users will get a best effort service, whereas others, such as news agencies, can buy a real-time service where data is due to happen within a specific timeframe.

SLA management was not natively implemented within the SocIoS framework, but was demanded to the QoSMONaaS application. To enable QoSMONaaS to monitor SLA compliance, the SLA have to be formalized according to the WS-Agreement language [12]. An excerpt of a possible SLA is provided in Fig.3. It describes two constraints, one for the end user and one for the service provider. Specifically, the "CallOfFindActivities" term expresses the constraint "the number of the FindActivities API invocations must be less than 5 in an observation windows of 20 seconds", while the "ResponseOffFindActivities" term expresses the constraint "the average response time of the FindActivities API must be less than 5 seconds in an observation windows of 20 seconds". To perform the integration between QoSMONaaS and the SocIoS framework two steps were necessary: i) data format adaptation; and ii) formalization of the domain description for the sake of the monitoring component.

Regarding the data format, it was necessary to convert SocIoS data to the format used by QoSMONaaS. QoSMONaaS data are formatted according to the Google Protocol Buffers (GPB) [11] format, that is a way of encoding structured data in an efficient yet extensible format. Concerning the domain description, an ontology describing the relationships between the data concepts represented in the SocIoS data stream was created. As an example Fig. 4 shows the definition of the “FindActivities” concept. Such a concept is associated to the call of the SocIoS API `findActivities()` which retrieves the activity stream of a specified SN user. The concept is described by using the property “MultipleFilter” and the property “pr-OnStream”. “MultipleFilter” is used to explain how to identify a particular concept on the data stream (in this case a filter with multiple conditions is used), while “pr-OnStream” identifies the stream where this kind of concept can be found. This property is useful in case of multiple input streams. It is worth noting that thanks to the high modularity of QoSMONaaS, its adoption for the monitoring of the services provided by the SocIoS framework does not require any modification of the application, but only the provisioning of proper configuration data.

In order to demonstrate the integration between QoSMONaaS and the SocIoS framework we have executed a simple run with a controlled workload to check the correct operation. The selected scenario involves three actors, namely an end user of the SocIoS platform, a service provider (SocIoS) and QoSMONaaS that is in charge of monitoring the SLA signed by the user and the service provider. Before starting the monitoring of the process, two actions are required: i) Provider Registration, and ii) User Registration. During the first step the SocIoS service provider has to register to the QoSMONaaS application. At that time the SLA and the ontology are provided to QoSMONaaS for the automatic generation of the state machines allowing the actual monitoring of the SLA compliance based on the data provided by the SocIoS framework. After the registration the provider enables end users to monitor the provided service. During the second step, the end user registers to QoSMONaaS for monitoring the SocIoS-based service, and specifies a monitoring time window. Once the first two steps are completed the monitoring task can start and the end user is informed in real-time about violations of KPIs in accordance with the agreed SLA.

In the SocIoS use case the SLA regulates the relationship between the Core Services and the end users. The Core Services deliver an API to the end users either implicitly (if the end user employs a front-end) or explicitly (if the end user is a developer). The important aspects that need to be guaranteed in order to ensure a certain level of provided quality from the Core Services are the amount of API calls available (availability) and the response time for each one of these calls. The reason is that during a pilot execution phase it has been noticed that the dependency of the Core Services to the underlying SNS APIs affects the quality as perceived by the end user (Quality of Experience-QoE). The SLA to be monitored includes two metrics for the `findActivities()` SocIoS API call, i.e. `FindActivities` and `ResponseOfFindActivities`. The first refers to the number of calls made to that Core Service method, whereas the latter to the time it took the Core Services server to respond.

```

<wsag:GuaranteeTerm Name="CallOfFindActivities" Obligated="User">
<wsag:ServiceScope>
<wsag:ServiceName>Metering</wsag:ServiceName>
</wsag:ServiceScope>
<wsag:ServiceLevelObjective>
<wsag:KPITarget>
<wsag:KPIName>FindActivities</wsag:KPIName>
<wsag:Target>Count IS_LESS 5 per 20 seconds</wsag:Target>
</wsag:KPITarget>
</wsag:ServiceLevelObjective>
</wsag:GuaranteeTerm>

<wsag:GuaranteeTerm Name="ResponseOfFindActivities" Obligated="Provider">
<wsag:ServiceScope>
<wsag:ServiceName>Metering</wsag:ServiceName>
</wsag:ServiceScope>
<wsag:ServiceLevelObjective>
<wsag:KPITarget>
<wsag:KPIName>ResponseOfFindActivities</wsag:KPIName>
<wsag:Target>AVG IS_LESS 5000 per 10 seconds</wsag:Target>
</wsag:KPITarget>
</wsag:ServiceLevelObjective>
</wsag:GuaranteeTerm>

```

Fig. 3 An example of SLA for the SocIoS framework

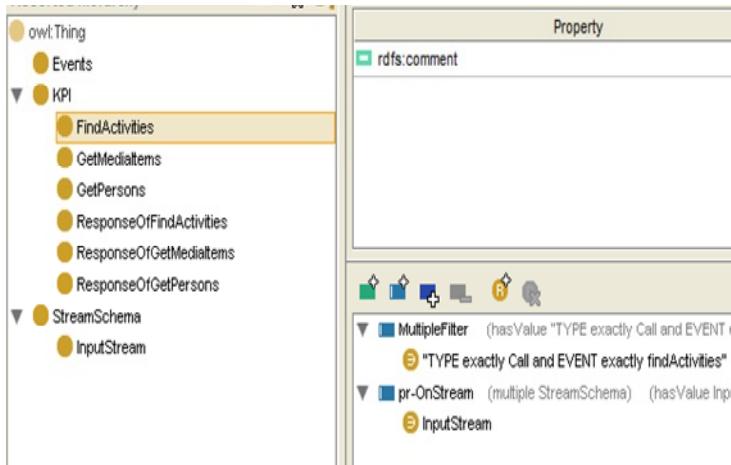


Fig. 4 Ontological description of the SocIoS framework domain

In order to feed QoSMONaaS a log of the SocIoS pilots' execution and evaluation phase has been used, in which the values of interest were recorded in the form shown in Fig. 5.

```
# | 2013-02-04
T12:38:23.359+0200|INFO|glassfish3.1|
javax.enterprise.system.std.com.sun.enterprise.server.logging|_ThreadID=104;
_ThreadName=Thread-1;| SOCIOS_EVAL CORE
class eu.sociospesproject.sociospesapi.server.adaptors.youtube.YoutubeAdaptor
calling findActivities() and getting 16 results in 4320 msec|#]
```

Fig. 5 SocIoS sample data

An instance that depicts the violations that occurred during the pilot evaluation while the end-user were invoking the *findActivities()* method is illustrated in Fig. 6.

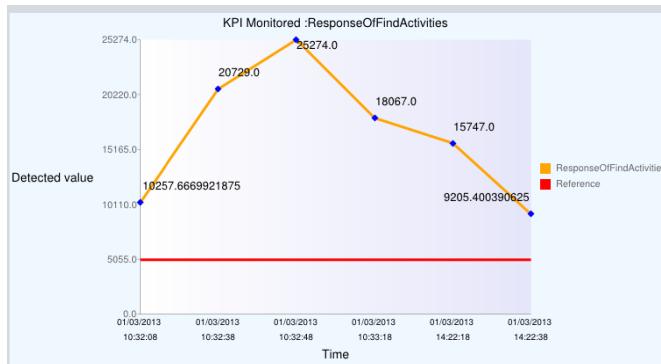


Fig. 6 Outcome of the QoSMONaaS component for the SocIoS SLA evaluation

Details about the two metrics that were evaluated and the SLA breaches can be seen in Fig. 7 and Fig. 8 for the amount of API calls in the time unit (availability) and the response time of these calls, respectively.



Fig. 7 Violation report for the number of invocations of *findActivities* in a time unit

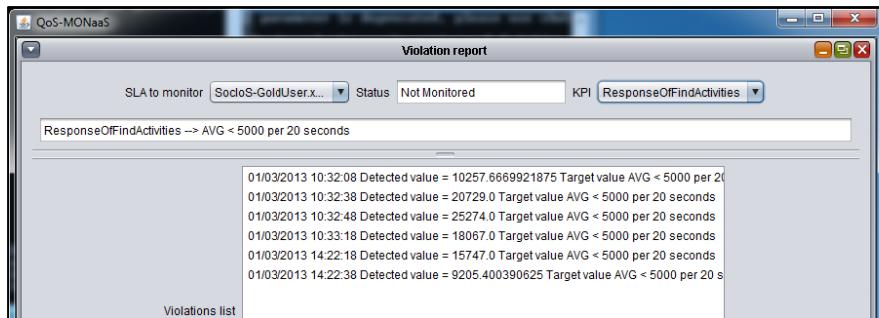


Fig. 8 Violation report for the response time of findActivitiesmethod in a time unit

5 Conclusions and Future Work

In this paper we have discussed a joint work made by partners of two EC funded projects, namely the SRT-15 and the SocIoS projects. In particular we have described the usage of SRT-15 component for QoS monitoring, named QoSMONaaS, for the Quality of service assessment of a typical SocIoS application. To demonstrate the integration between QoSMONaaS and SocIoS we have executed a simple run with a controlled workload to check the correct working. As future work a more extensive experimental campaign will be performed in order to validate the proposed integrated approach. Furthermore, security issues affecting QoS monitoring application for cloud computing environment will be investigated [13].

Acknowledgments. This work has been partly funded by the European Commission's 7th Framework Programme under contract no.257774 and no. 257843. The authors also wish to thank all their project co-workers, in particular Emmanuel Sardis and Luigi Sgaglione.

References

1. The SocIoS project, <http://www.sociosproject.eu/>
2. The SRT-15 project, <http://www.srt-15.eu/>
3. Romano, L., De Mari, D., Jerzak, Z., Fetzer, C.: A Novel Approach to QoS Monitoring in the Cloud. In: 2011 First International Conference on Data Compression, Communications and Processing (CCP), June 21-24, pp. 45–51 (2011)
4. Adinolfi, O., Cristaldi, R., Coppolino, L., Romano, L.: QoS-MONaaS: A Portable Architecture for QoS Monitoring in the Cloud. In: 2012 Eighth International Conference on Signal Image Technology and Internet Based Systems (SITIS), November 25-29, pp. 527–532 (2012)
5. Cicotti, G., D'Antonio, S., Cristaldi, R., Sergio, A.: How to Monitor QoS in Cloud Infrastructures: The QoSMONaaS Approach. In: Fortino, G., Badica, C., Malgeri, M., Unland, R. (eds.) Intelligent Distributed Computing VI. SCI, vol. 446, pp. 255–264. Springer, Heidelberg (2012)

6. Cicotti, G., Coppolino, L., Cristaldi, R., D'Antonio, S., Romano, L.: QoS monitoring in a cloud services environment: the SRT-15 approach. In: Alexander, M., et al. (eds.) Euro-Par 2011, Part I. LNCS, vol. 7155, pp. 15–24. Springer, Heidelberg (2012)
7. Tserpes, K., Papadakis, G., Kardara, M., Papaoikonomou, A., Aisopos, F., Sardis, E., Varvarigou, T.A.: An Ontology for Social Networking Sites Interoperability. In: Proc. of the 4th International Conference of Knowledge Engineering and Ontology Development (KEOD 2012), pp. 245–250 (2012)
8. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.: The Many Faces of Publish/Subscribe. ACM Computing Surveys (CSUR) 35(2), 114–131 (2003)
9. Complex Event Processing: Applications, products, research, and developments in event processing, <http://www.complexevents.com/event-processing/>
10. Ying, X., Pan, K., Wu, X., Guo, L.: Comparisons of randomization and K-degree anonymization schemes for privacy preserving social network publishing. In: Proceedings of the 3rd Workshop on Social Network Mining and Analysis, SNA-KDD 2009 (2009)
11. Google Protocol Buffer,
<https://developers.google.com/protocol-buffers/>
12. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement). Global Grid Forum 2 (2004)
13. Ficco, M., Rak, M., Di Martino, B.: An intrusion detection framework for supporting SLA assessment in Cloud Computing. In: Proceedings of the 2012 4th International Conference on Computational Aspects of Social Networks, CASoN 2012, pp. 244–249 (2012)

Personalized Recommendation of Semantically Annotated Media Contents

Alba Amato, Beniamino Di Martino, Marco Scialdone, and Salvatore Venticinque

Abstract. The optimal decision about the best set of recommendations to the user is a relevant problem in different application contexts. Here we focus on a methodology for dynamically and efficiently retrieval and delivery of multimedia contents, like documents, images, video etc., which have been annotated with semantic information. We introduce the definition and the computation of the similarity between sets of concepts belonging to a common ontology and discrete optimization technique for choosing the ones to be recommended. We enable personal agents reasoning about the best set of recommendations, which are relevant to the user's profile, so optimizing his/her satisfaction. We present its implementation in a framework that supports experts in the domain of the Cultural Heritage to augment the archaeological site with a set of multimedia contents.

1 Introduction

The definition and the computation of the similarity between sets of concepts belonging to a common ontology is really important to help sharing knowledge in those contexts where documents, images, video etc. are annotated with semantic information. Similarity-based classification is useful for retrieving and filtering information in an automatic way when it needs to recommend any contents or applications which are relevant to a profile that has been described by the same ontology. In fact the similarity between two sets of concepts represents the evaluation of the similarity of the information content they share in the ontology. This is essential in application aimed at managing knowledge and filtering it in such a way that all

Alba Amato · Beniamino Di Martino · Marco Scialdone · Salvatore Venticinque

Department of Industrial and Information Engineering,
Second University of Naples, Naples, Italy

e-mail: {alba.amato,beniamino.dimartino,
salvatore.venticinque}@unina2.it,
marcoscialdone@hotmail.com

the relevant information for the user is returned in a reasonable amount of time and in an automatic way. One important application field is represented by Agent Systems, in particular for improving the decision of the agents used for dynamically and efficiently data processing, retrieval and delivery. In this paper, we introduce a method for enabling personal agents reasoning about the best set of recommendations, which are relevant to the user's profile, so optimizing his/her satisfaction. We present its implementation in a framework that supports experts in the domain of the Cultural Heritage [1] to augment the archaeological site with a set of multimedia contents. The proposed methodology allows for evaluating the relevance of media contents, for filtering the ones which do not satisfy some constraints, and finally for decision about the best set to be recommended to the user. The paper is organized as follows: in Section 2 some relevant works are presented, Section 3 describes the application context; in Sections 4 is described the proposed solution and its application to a case study; in Sections 5 an experimental evaluation is presented; conclusions are drawn in Section 6.

2 Related Work

Several approaches have been developed to compute the similarity between sets of concepts using different methodologies. In this paper we take into account only those based on the distance between concepts. In particular, in the selected approaches, the similarity function is based on the idea that concepts which are close according to their positions in the ontology are more similar than topics that have a larger distance. In [2] authors chose the minimal linking approach based on the set of linkings M^l between concept sets A and B. For $r \in M^l$ it holds $\forall a \in A \ \forall b \in B$ with $(a, b) \in r$ and vice versa. For the distance calculation those elements from M^l is chosen for which the sum of concept distances Δ^* is minimal:

$$dist^l(A, B) = \min_{r \in M^l} \left(\sum_{(a,b) \in r} \frac{\Delta^*(a,b)}{|r|} \right)$$

In [3] authors assume that, given two sets of concepts A and B, it is not sufficient that a concept in the source set matches a concept in the target set to obtain 1 as result but it is necessary that all the concepts in the source set should have a related concept in the target set. Besides they believe that it is necessary a score to obtain a similarity function that respects the inequality that can occur if there are many concepts with high similarity in the two sets or if there are many concepts with low similarity in the two sets of concepts. So they introduce an algorithm based on the following formula:

$$\sqrt[m]{\frac{\sum_{k=0}^n (a_k)^m}{n}}$$

where a_k represents the similarity between a concept of the set A and the set B calculated using an extension of Dijkstra's algorithm; n represents the number of elements of the set A; m represent a score to specify how much the higher values

should be weighted with respect to lower ones. In [4] the distance between the sets A and B, is calculated as:

$$\sum_{c \in A, c' \in B} D(c, c')$$

where $D(c, c')$ is the length of the minimal path between the nodes corresponding to c and c' in the Ontology, if such a path exists, and is 0 otherwise. As this definition involves some redundancy, authors introduce the notion of normalized set of simple concepts. In [5] the similarity measure between the set A and B is calculated as:

$$SF(A, B) = \frac{1}{|A|} \sum_{t_i \in A} \max_{t_j \in B} S(t_1, t_2)$$

where:

$$S(t_1, t_2) = \begin{cases} e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}}, & \text{if } t_1 \neq t_2, \\ 1, & \text{otherwise} \end{cases}$$

where l is the length of the shortest path between concept t_1 and t_2 in the graph, h is the level in the tree of the direct common subsumer from t_1 and t_2 , $\alpha \geq 0$ and $\beta \geq 0$ are parameters scaling the contribution of shortest path length l and depth h , respectively. The depth of the direct common subsumer is used in the calculation to express the fact that topics at upper layers of hierarchical semantic nets are more general and are semantically less similar than topics at lower levels. Our approach is really similar to the previous but the scaling of the contribution of shortest path is computed as a normalization respect to the number of arcs between concepts.

3 The Agent Based MARA Framework

The availability of personal devices can be used to plan and support the tourist by suggesting him the itineraries, the Points Of Interest (POI) and by providing multimedia contents in the form of digital objects which can semantically augment the perceived reality. In this context relevant issues are the profiling of the user, the discovery and the delivery of the contents that can improve the user's satisfaction, new models of interactions with reality. The MARA (Mobile Augmented Reality in Archeology) project aims at designing and implementation of a context-aware platform for assisted fruition through mobile devices of archaeological sites. Agents execute autonomously and proactively in order to support the user's activity within the environment where he is moving. Agents discover surrounding objects, they use them to update the representation of the user's knowledge, react using the local knowledge to organize and propose the available contents and facilities by an interactive interface. They implement context aware services which use personal devices to collect perceptions and for content delivery. An ontology has been designed to describe the sites of interest and to annotate the related media. A general part includes the concepts which are common to all the class of applications that can be

modeled according to the proposed approach. Among the others the *Time* class and his properties (*CurrentTime*, *AvailableTime*, *ElapsedTime*, *ExploitationTime*) allow to organize and assist the visit taking into account time information and handling time constraints. *Position* class and its properties allow to localize the user and objects around him. An application specific part of the ontology, developed by the experts of the application field, includes the concepts that belong to the domain of the cultural heritage and additional classes and individual. The ontology is used for annotating the multimedia contents. To annotate texts, images and any kind of contents we extended the AktiveMedia tool. In Figure 1 a picture of the Amphitheater of S. Maria Capua Vetere is annotated with the *Column* and the *Arc* classes which are part of this kind of building. The output produced by the annotator is an RDF file with concepts and properties of the AktiveMedia ontology and of the domain ontology. Multimedia contents are automatically stored into the repository after the annotation phase. Each content can be linked to more points of interest.

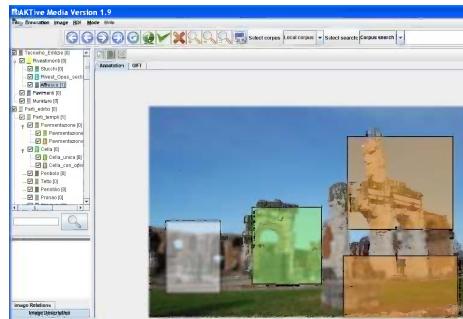


Fig. 1 The annotator

4 Context Aware Semantic Discovery

The semantic discovery service of MARA returns a set of digital objects related to POIs in the pervasive environments. Each content is annotated by concepts from the ontology and can be discovered by a SPARQL query to the content repository. The result of the query is a set of N instances of digital objects whose relevance to the user context has not been considered yet. The context awareness of the services is exploited by computing the relevance of each content to the profile \mathbf{p} , which is that semantic representation of the context, including interest, position, etc.. The annotation and the user's profile are represented using the Vector Space Model (VSM). VSM is a model for semantic representations of contents as vectors of items created by G. Salton [7]. In our case \mathbf{a}_j is the vector of concepts $c_{i,j}$ of the domain ontology.

- $\mathbf{a}_j = \langle \{c_{1,j}, o_{1,j}\}, \{c_{2,j}, o_{2,j}\}, \dots, \{c_{l,j}, o_{l,j}\} \rangle \forall j = 1..N$
- $\mathbf{p} = \langle c_1, c_2, \dots, c_m \rangle$

Sizes l and m are the number of different concepts that appear in the annotation \mathbf{a}_j and in the profile \mathbf{p} . If a term occurs in the annotation, $o_{k,j}$ is the number of

occurrences of that term in the annotation. We defined a score $A(\mathbf{a}_j, \mathbf{p})$ to measure the relevance of an annotation \mathbf{a}_j to the profile \mathbf{p} by the following formula 1

$$w_j = w(\mathbf{a}_j, \mathbf{p}) = \frac{1}{l} \sum_{k=1}^l r_k \text{ where } r_k = o_{k,j} * \frac{1}{m} \sum_{i=1}^m \frac{1}{d_{k,i} + 1} \quad (1)$$

where $d_{k,i}$ is the minimum number of edges that connect the node representing the concept $c_{k,j}$ of the annotation to c_i of the profile. In Equation 1, for each item $c_{k,j}$ of the vector \mathbf{a}_j the relevance to the profile \mathbf{p} is computed by adding the relevance of that concept to each concept of the profile, and by multiplying each contribution for the number of occurrences $o_{k,j}$. The relevance between two concepts is calculated by dividing 1 by the number of edges of the ontology that connect the node representing the concept c_k of the profile to $c_{k,j}$ plus 1. As a result we have a score for each annotated item that is associated to a POIs so that it is possible to order the items and the POIs according to user's preferences. However a user have some additional limitations that are the time available, the distance he wants to go away, the device's capability to play certain types of content, etc. For this reason it is necessary to select the valid contents excluding, for example, those that cannot be enjoyed by his device. Besides it is necessary that the duration of the visit does not exceed the time that is available to the user. To do this we formulate a set of constraints to take into account the capabilities of the device and the time available to play the contents. Hence, the best set of contents to be delivered should 1) maximize the score, 2) be compliant with the device technology and 3) not exceed time and space limits. This problem can be reduced to a *discrete optimization problem* that consists in searching the optimal value (maximum or minimum) of a function $f : \mathbf{x} \in \mathcal{X}^n \rightarrow \mathcal{R}$, and the solution $\mathbf{x} = \{x_1, \dots, x_n\}$ in which the function's value is optimal. $f(\mathbf{x})$ is said *cost function*, and its domain is generally defined by means of a set of m constraints on the points of the definition space. Constraints are generally expressed by a set of inequalities:

$$\sum_{i=1}^n b_{i,j} x_i \leq a_j \quad \forall j \in \{1, \dots, m\} \quad (2)$$

and they define the set of feasible values for the x_i variables (the *solutions space* of the problem). In our case: $w_i \geq 0 \forall i = 1..N, W = \{\mathbf{b}_1, \dots, \mathbf{b}_m\}$ where w_i represents a score and B a set of constraints \mathbf{b}_i , each one composed of $N+1$ integer. We have to compute

$$\max \sum_{k=1}^N w_k x_k$$

so that

$$\sum_{k=1}^N b_{i,j} x_k \leq b_{j,N+1}$$

with $x_i \in \{0, 1\} \forall i = 1..N$. The goal is to maximize the value delivered. The vector \mathbf{x} represents a possible solution: its components x_i are 1 or 0 depending on whether the object is included or not in the best set. To set the constraints we create a ta-

ble with a column for each content and a row for each requirement. The rows of the matrix B are contained in each cell $b_{i,j} = 1$ if the requirement is necessary for the content in that column, $b_{i,j} = 0$ otherwise. In Table 4 we specify the ability of the device to reproduce audio, to reproduce video, to show text files, to show 3D models. Furthermore in the last column of the table we have $b_{i,N+1} = 0$ if the device cannot play/display the content to which the row i refers, N otherwise. Table 4 (a) represents the case of a device that can play audio and video and show text file, but it cannot show 3D contents. For example, the fourth row represents the fact that the device can not show 3D contents, so it is necessary to exclude the content C_2 . This example, where $b_{i,j} \in \{0, 1\}$, is a particular case of a more general problem of greater complexity can be formulated by representing with $b_{i,j} \in \mathbb{Z}$ the quality of a content and with $b_{i,N+1}$, the degree of a capability for the device. In this case a feasible solution can include a subset of contents which are compliant with the quality of service that can be perceived by the device. Similarly, time constraints are set using a vector having for each content the minutes needed for its fruition. The sum of the duration of each content included in the solution must be less or equal than user availability (in terms of minute for the visit). For example, if the visitor can spend 30 minutes for the visit and the minutes needed for each content are the one specified in Table 4 (b). To satisfy the constraint: $\sum_{i=1}^N b_i x_i \leq 30$, we must include in the solution C_3 only without other content. Of course it means that this solution is feasible, but it does not mean that this is the best one. To address the complexity of this problem we use the Branch and Bound technique described in [6]. After that a solution is available, referring each content to a POIs, the itinerary is computed in real time by a Navit facility that allows to find the best route through all the destinations.

Table 1 Constraints and device capabilities

	(a) Content requirements						(b) Time requirements				
	C_1	C_2	\dots	C_n	Device		C_1	C_2	\dots	C_n	Available
Audio	1	1	\dots	0	N						
Video	1	0	\dots	1	N						
Pdf	1	1	\dots	1	N						
3D	0	1	\dots	0	0						

5 Experimental Results

In this section we evaluate effectiveness and performance of the proposed approach in order to verify that recommendation are really relevant and are provided in real time to the user. We evaluate the similarity between a given profile and a set of contents, which have been expressed by a vector of concepts belonging to our ontology. After that in a second step we choose only the set of contents that maximize the relevance, but satisfy a number of constraints defined in the user profile. We used a machine with Intel Core i5 2.67GHz and 4GB RAM. The first test has been performed varying 1, 2, 4, 8, 16 and 32 concepts for describing both content and and

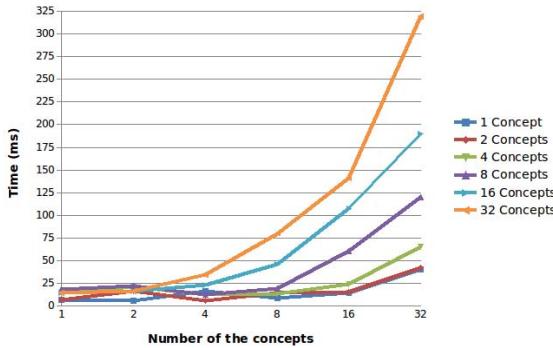


Fig. 2 Performance evaluation of similarity with random profile and contents

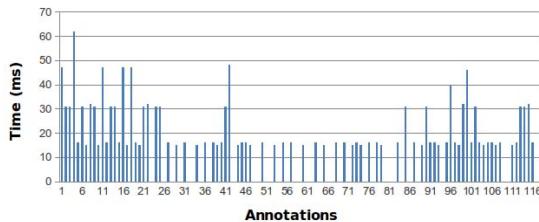


Fig. 3 Time to compute the similarity for all annotations of KB

profile. In this case concepts have been chosen randomly and only performance are significant. In Figure 2 we can see that the time needed for computing the similarity increases exponentially with the number of concepts. In a second test, in order to define a significant profile we suppose that it is automatically updated by the application that saves the most recent 16 distinct concepts belonging to the last viewed contents. We have built in this way a user profile keeping an annotation with 16 concepts and we compared it with all the others in our knowledge base. The knowledge base is composed of 118 contents produced and annotated by experts of the application domain, who are also authors of the ontology. The number of concepts used for each annotation varies from 2 to 19. We observe in Figure 3 that the mean time is 15.5 ms, which means that 2 seconds are enough to update the similarity between the user profile and all contents in our knowledge base. In the second step we run the Branch and Bound algorithm for filtering only the most relevant contents which satisfy the user/device constraints. For each content we specify the time needed for delivery, and if they require to show text, to play audio or video. The computation has been performed with different input and varying the number of contents till 32. We can see in Figure 4 that here the time increases very fast after a certain number of contents till to 22.5 seconds. We can consider that a reasonable choice would be

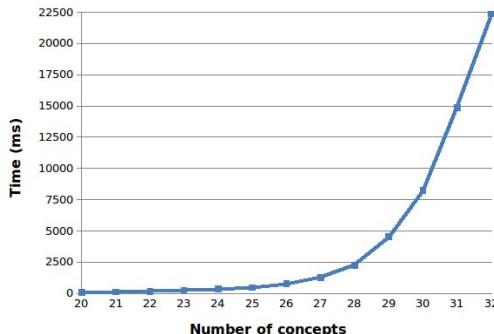


Fig. 4 Performance evaluation of content filtering

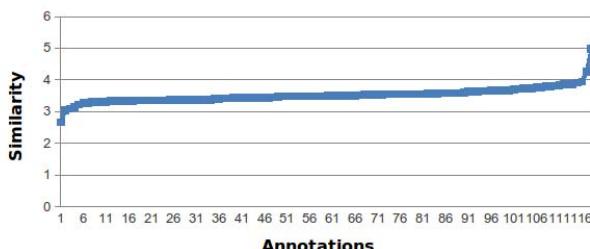


Fig. 5 Distribution of similarity

to include no more than 26 items in this step. In fact in this case the time does not exceed 1 second and it is comparable with that one needed to evaluate the similarity. Moreover it would be ineffective to consider not relevant contents and to exceed with the number of recommendations. Unfortunately the evaluation of similarity does not provide good results. In Figure 5 we show the similarity between the user profile and all the annotations in the knowledge base in ascending order. The problem we experienced is a narrow range between the less relevant content and the most relevant one equals to 30%. It is due above all to the limited depth of the ontology and the recurrence of the same concepts. To confirm the previous consideration we also generated random annotations by the ontology varying the maximum distance between the a concept of the annotation and the other concepts of the profile. This allowed us to evaluate how the defined metric and the ontology affect the similarity regardless how the annotations have been produced. However in Figure 6 we noticed the same results. We can conclude that with this kind of ontology, even if the proposed methodology could be used to compute recommendations in real time, it needs to improve the metric for evaluating the relevance of annotations. In order to get a wider range within which the similarity can varies we can consider to use weights for defining the relevance of concept in the annotation, in the profile or in the ontology.

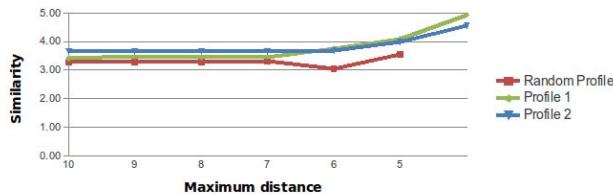


Fig. 6 Second Experiment with Random Profile

6 Conclusion

In this paper we propose the study of semantic techniques for computing the similarity between sets of concepts belonging to the same ontology, representing user's profile and annotation of multimedia contents. We evaluated this implementation in a framework that supports the experts in the domain of the Cultural Heritage to augment the archaeological site with a set of multimedia contents. We discussed performance results and effectiveness of the proposed techniques to recommend such media to visitors. Even if performance figures demonstrate the feasibility of the approach and they can be used for tuning applications and dimensioning the computing resource, however the chosen metric is not effective to evaluate the relevance of annotations with shallow ontologies. Future work includes the design of solution to the described problems by the associations of weights to the concepts into the annotation and into the user profile. We will also consider the extension of the semantic matching approach by computing the semantic distance between concepts belonging to different ontologies.

Acknowledgements. This work has been supported by PRIST 2009, *Fruizione assistita e context aware di siti archeologici complessi mediante terminali mobili* and by FP7- 608806, *CoSSMic: Collaborating Smart Solar-powered Micro-grids*, FP7-SMARTCITIES-2013.

References

1. Amato, A., Di Martino, B., Venticinque, S.: Semantically augmented exploitation of pervasive environments by intelligent agents. In: Proc. of 10th International Symposium on Parallel and Distributed Processing with Applications, pp. 807–814. IEEE CPS (2012)
2. Billig, A., Blomqvist, E., Lin, F.: Semantic Matching based on Enterprise Ontologies. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 1161–1168. Springer, Heidelberg (2007)
3. Cordi, V., Lombardi, P., Martelli, M., Mascardi, V.: An Ontology-Based Similarity between Sets of Concepts. In: WOA (2005)
4. Bouquet, P., Kuper, G., Scoz, M., Zanobini, S.: Asking and answering semantic queries. In: Proc. of Meaning Coordination and Negotiation Workshop (2004)

5. Haase, P., Siebes, R., van Harmelen, F.: Peer Selection in Peer-to-Peer Networks with Semantic Topologies. In: Bouzeghoub, M., Goble, C.A., Kashyap, V., Spaccapietra, S. (eds.) ICSNW 2004. LNCS, vol. 3226, pp. 108–125. Springer, Heidelberg (2004)
6. Aversa, R., Di Martino, B., Mazzocca, N., Venticinque, S.: High Performance Computing: Paradigm and Infrastructure A Hierarchical distributed shared memory Parallel Branch & Bound Application with Pvm and OpenMP for multiprocessor clusters. Wiley (2004)
7. Salton, G., Lesk, M.E.: Computer evaluation of indexing and text processing. *Journal of the ACM* 15(1), 8–36 (1968)

Supporting Cloud Governance through Technologies and Standards

Victor Ion Munteanu, Teodor-Florin Fortiș, and Adrian Copie

Abstract. With the dawn of Cloud computing, competitors all over the world rush to take advantage of this new paradigm and the economic models it brings. Unfortunately for the earnest, cloud computing is not short of problems, most of which are being addressed through different platform-as-a-service, cloud management or governance solutions. Coming to support a natural integration between cloud management and cloud governance, CloudML and other specifications like Tosca enable a model-based approach which addresses inconsistencies found at infrastructure or platform levels thus contributing to the automation of the cloud service lifecycle.

1 Introduction

Cloud computing is a new paradigm which brings about new ways of thinking about and using resources, by exposing them as services and enabling a secure, flexible and scalable use, all while being affordable.

In the context of this new *cloud economy*, more and more small and medium-size enterprises (SMEs) are moving to the cloud in an effort to reduce costs by accessing the shared infrastructure as well as existing automation processes, which enables them to maintain less of their own personnel and infrastructure.

Despite these clear advantages brought by the cloud, there are also downfalls which lead to poor exploitation of this environment. One of the early ones is *vendor-lockin* which locks the cloud developer to a specific cloud provider. Other problems relate to lack of transparency regarding security and privacy practices among cloud providers [11].

Victor Ion Munteanu · Adrian Copie
West University of Timișoara, bvd. V.Pârvan 4, Timișoara, Romania
e-mail: {vmunteanu, adrian.copie}@info.uvt.ro

Teodor-Florin Fortiș
Institute e-Austria Timișoara, bvd. V.Pârvan 4, Timișoara, Romania
e-mail: fortis@info.uvt.ro

In order to tackle these problems and increase efficiency, cloud developers must resort to additional tools in order to provide cross-cloud interoperability, application deployment and task automation, tools like PaaS solutions or cloud management ones.

Cloud management, even though being close to existing PaaS solutions, focuses on providing specific cloud management tasks as detailed in [7, 9]. Its primary role, along with that of PaaS solutions, is to enable an easier cloud adoption by delegating and automating cloud computing operations like *resource provisioning, monitoring, application deployment, SLA management*, etc.

Complementary to the services provided by cloud management, cloud governance focuses on providing integration at SaaS level by providing mechanisms that enable service ecosystems through the specification of policies which are then carried out by the underlying cloud management solutions.

In their papers, Distributed Management Task Force (DMTF) offers a reference architecture for cloud governance along with its interactions with cloud management [8]. Other reference models which identify the role it plays are presented in [19, 23].

One key element for a governance-management solution consists by the Service Level Agreements (SLAs) which, along with Service Level Objectives (SLOs) and Service Level Agreement Templates (SLATs), provide support for cloud monitoring at all layers (*aaS). Having autonomic detection of SLA violations is crucial as “*prevention of SLA violations avoids unnecessary penalties providers have to pay in case of violations [...] interactions with users can be minimized*” [10].

This paper focuses on cloud governance, specifically the way in which existing technologies and standards interact in order to support it and where none exist and there is room for future development.

The rest of the paper is structured as follows. Background information is discussed in Section 2. Section 3 covers the service lifecycle and shows relations between existing technologies which interact at different steps. Finally, conclusions and future work are covered in Section 4.

2 Background Information

2.1 Cloud Governance and Service Lifecycle

Stemming from Service Oriented Architecture (SOA) governance and ISO/IEC 38500 principles for IT governance, cloud governance “*focuses on the creation, communication and enforcement of service policies [...] that consist of a set of constraints and capabilities that govern how services and their consumers interact*” [20].

Through these service policies, cloud governance has the ability to manage and provide a comprehensive cloud service lifecycle which, in turn, enables cloud developers ease-of-use when developing applications [12]. This ability is essential for the cloud as providing self-managed automated service lifecycle reduces costs and complexity of the tasks developers have to perform.

While SOA service lifecycle has been covered extensively in the literature [14, 21, 15], research into cloud service lifecycle is still in its early stages, the work done focusing on aspects related to both service distribution and scaling [8, 17].

Moreover, current standardization directions related to cloud services and governance revolve around OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA)¹. It enables the description of services across all *aaS layers through the use of service templates. Additionally, TOSCA builds upon and provides integration with other standards such as DMTF OVF², DMTF CIM³, DMTF CIMI⁴, OGF OCCI⁵ and OASIS SCA⁶.

2.2 *Cloud Management and Cloud Automation*

Complementary to cloud governance, cloud management has an executive approach providing efficiency and ease of use in the utilization of cloud resources, thus ensuring that the enterprise objectives set by the governance are achieved through planning, deploying, running and monitoring current activities.

A typical cloud management solution usually revolves around the Monitor-Analyse-Plan-Execute cycle performed around a cloud resource Knowledge base, as detailed in IBM's MAPE-K model [18].

For complete management to take place, automation along with compliance, security, fault tolerance and recovery etc. must be offered by a cloud management solution. Only them will it meet “*programmer expectations in regard to resource management for scalability, data durability, and high availability*” [1].

New attempts to address cloud management come as model-driven approaches where applications are modeled based on their components by emphasizing the relationships between them. CloudML is one such approach which aims at having a cloud modeling language for cloud application development [5].

Current standards that are important to cloud management and cloud automation include OGF OCCI (protocol and API for management tasks) and DMTF CIMI (runtime maintenance and provisioning of cloud services through REST / HTTP).

2.3 *Cloud Governance Requirements for Cloud Modeling*

Typical cloud application modeling spans from development to deployment, from defining the model for the application architecture to defining the way it will be monitored and scaled.

¹ <http://www.oasis-open.org/committees/tosca/>

² <http://www.dmtf.org/standards.ovf>

³ <http://www.dmtf.org/standards/cim>

⁴ http://dmtf.org/sites/default/files/standards/documents/DSP0264_1.0.0.pdf

⁵ <http://occi-wg.org/>

⁶ <https://www.oasis-open.org/committees/sca-j/>

Modeling can be split into design time modeling and run time modeling, both being essential for existing cloud management solutions as design time modeling describes the general structure of the application and the relations it has with the underlying infrastructure (provisioning and deployment at IaaS) while run time modeling enables the component grouping and scaling based on specified QoS constraints.

In the case of cloud governance, modeling is closely related to the service life-cycle, at design time the focus being on specifying existing connections with other services (service dependence), modeling offers, pricing etc while at run time it focuses on the modeling of service QoS and scalability.

2.4 Multi-agent Approach to Cloud Management and Governance

Due to their nature, multi-agent systems make perfect candidates for distributed environments as “*they offer the high-level software abstractions needed to manage complex applications and because they were invented to cope with distribution and interoperability*” [2].

Unfortunately, the number of agent based implementation in cloud computing is relatively small even though “*the convergence of interests between MASs that need reliable distributed infrastructures and cloud computing systems that need intelligent software with dynamic, flexible, and autonomous behavior will result in future intelligent services and applications*” [22].

One of the more notable multi-agent systems in charge of cloud management is mOSAIC’s Cloud Agency which offers functionality for the provisioning, monitoring and reconfiguration of cloud resources for major cloud vendors like Amazon Web Services, OpenNebula, GoGrid and others [24, 25, 26].

Another solution uses self-organizing agents in order to provide automated cloud service composition [13] and deals with dynamic provisioning of cloud resources while having incomplete knowledge about them.

A multi-agent architecture specifically designed to handle cloud governance while encompassing an underlying cloud management solution is discussed in [12, 16].

3 Cloud Service Lifecycle

As identified in [17], the main operations pertaining to the cloud service lifecycle are:

Design-time operations – refer to general information about the services which enable it to be published.

Provisioning operations – refer to the process of discovering and contracting services.

Deployment operations – refer to the process of instantiating and commissioning services.

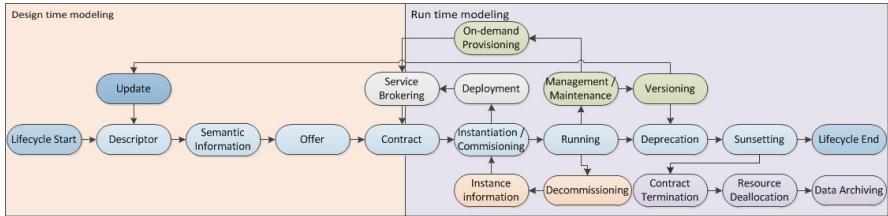


Fig. 1 Service lifecycle in contrast to cloud modeling

Execution operations – refer to the process of managing a running instance along with any economic implications it might have.

Retirement operations – refer to service retirement, ending its lifecycle

Figure 1 shows the above expanded service lifecycle in close relation with cloud modeling.

Related work covering the cloud service lifecycle in [6] identifies similar operations (phases): definition phase, offering phase, subscription and instantiation phase, production phase.

Moreover, [3] shows how TOSCA interacts within the service lifecycle through operational aspects (deployment, termination and management of a service) and the definition of service templates thus providing reusability and automation of service management.

While not running at SaaS layer, CloudML, as a Domain-Specific language (DSL) for cloud provisioning, addresses a series of challenges which are part of the service lifecycle like complexity (endpoints, frameworks, topologies), multi-clouds (working with several cloud providers), reproducibility (reusable templates and models), shareability (sharing of templates and various files), robustness (making use of existing technologies) and metadata dependency (provisioning models etc) [4].

Somehow different, Cloudify⁷ is an open source PaaS stack which features cloud management and basic cloud governance through service lifecycle support. It enables the definition of applications as a collection of services which have lifecycle operations (events) which can be customized and also enables the definition of metrics and thresholds which can be used for application scaling.

A multi-agent approach to handling this service lifecycle is covered in [12, 17], as well as the most important lifecycle activities are covered in [16].

3.1 *Choreography between Standards*

Many of the existing standards can be combined in order to cover different aspects needed to achieve successful cloud governance, as can be seen in Figure 2 where each color represents a different technology/standard as follows:

⁷ <http://www.cloudifysource.org/>

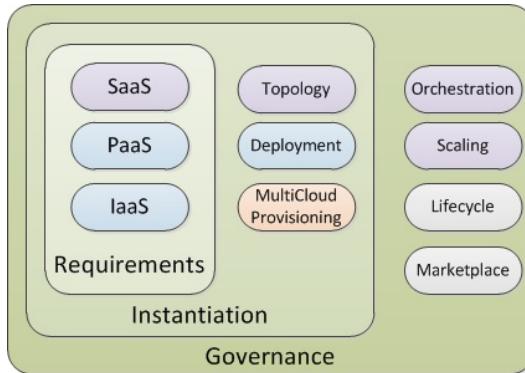


Fig. 2 Cloud governance layers

CloudML (blue) can be used to model requirements at IaaS and PaaS level as well as deployment specifications.

CIMI (red) enables communication with cloud providers for the provisioning and management of cloud resources.

TOSCA (purple) offers service description as well as topology, orchestration and scaling.

Grayed items are still not completely covered by any standard / technology.

Though instantiation items are divided, they can be handled by multiple standards as CloudML can handle provisioning through its runtime components as well as it can define the topology between various components.

3.2 Service Example

In order to exemplify the cloud service lifecycle, both elementary and composite services have been chosen in order to underline different aspects that exist in the lifecycle stages as shown in Table 1.

Each of these services has different requirements and offered capabilities. Following the description of these services we can determine that they can be divided into two groups:

Elementary – that only have IaaS and PaaS requirements. They are: Encryption, Geolocation, Image

Composite – have SaaS requirements. They are shown in Table 2 along with their dependencies.

In order to be able to orchestrate services, they need to be discovered based on capabilities, therefore requirements at SaaS level need to be expressed as capabilities.

Furthermore, when scaling takes place, all dependencies (and their dependencies) must be scaled as well.

Table 1 Example of elementary and composite services

Name	Description	Requirements
Identity management	Holds user identities and credentials	SaaS
Encryption	Performs on demand encryption, validation and signing	IaaS, PaaS
Geolocation	Handles information related to geolocation	IaaS, PaaS
Image	Handles image processing and storing	IaaS, PaaS
Geographic Information System	Handles all types of geographical data	SaaS
Micro blogging	Handles user micro blogs	SaaS
Email	Cloud email service	SaaS
Document	Handles the hosting, signing etc. of documents	SaaS
Weather forecasting	Cloud weather forecasting	SaaS
Communication	Enables conferencing over IP	SaaS
Financial	Handles invoices, billable time etc.	SaaS
Books	Online electronic book store	SaaS
Payment	Online payment service	SaaS
Shipping	Offers online shipment tracking	SaaS
Chatting	An online chatting service	SaaS

Table 2 Composite services and their dependencies

Name	Dependencies
Identity management	Encryption
Geographic Information System	Geolocation, Image
Micro blogging	Identity management, Image
Email	Identity management, Encryption
Document	Identity management, Encryption
Weather forecasting	Geographic Information System
Communication	Identity management
Financial	Identity management, Document, Payment
Books	Identity management, Financial, Document
Payment	Encryption
Shipping	Identity management, Geolocation
Chatting	Identity management, Communication

4 Conclusions and Future Work

This article presents a survey on cloud management and governance showing different aspects and the technologies and standards that address them.

In that regard, CloudML and TOSCA are discussed along with the implication they have in the cloud service lifecycle during design time modeling and run time modeling.

Furthermore, existing multi-agent solutions which cover cloud management and cloud governance are presented, underlining any support they give to the cloud service lifecycle.

Future work consists of full integration of CloudML and TOSCA within the existing multi-agent architecture while adding support for various technologies like Puppet or Chef for automatic cloud service deployment and configuration.

Acknowledgements. The research leading to these results has received partial funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 318484 (FP7-ICT 2011-8-318484 MODAClouds) and Romanian Government grant PN-II-ID-PCE-2011-3-0260 (AMICAS). The views expressed in this paper do not necessarily reflect those of the corresponding projects' consortium members.

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.: Above the clouds: A Berkeley view of cloud computing. Berkeley report (2009),
<http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf>
2. Bellifemine, F., Poggi, A., Rimassa, G.: Developing multi-agent systems with a fipa-compliant agent framework. Software-Practice and Experience 31(2), 103–128 (2001)
3. Binz, T., Breiter, G., Leymann, F., Spatzier, T.: Portable Cloud Services Using TOSCA. IEEE Internet Computing 16(03), 80–85 (2012),
doi: <http://doi.ieee.org/10.1109/MIC.2012.43>
4. Brandtzæg, E., Mosser, S., Mohagheghi, P.: Towards CloudML, a Model-based Approach to Provision Resources in the Clouds. In: Stoerle, H., Botterweck, G., Bourdelle, M., Kolovos, D., Paige, R., Roubtsova, E., Rubin, J., Tolvanen, J.-P. (eds.) Joint Proceedings of Co-located Events at the 8th European Conference on Modelling Foundations and Applications (ECMFA 2012), pp. 18–27. Technical University of Denmark, Copenhagen (2012)
5. Brandtzæg, E., Parastoo, M., Mosser, S.: Towards a Domain-Specific Language to deploy applications in the clouds. In: Proceedings of the Third International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2012), pp. 213–218 (2012)
6. Breiter, G., Behrendt, M.: Life cycle and characteristics of services in the world of cloud computing. IBM Journal of Research and Development 53(4), 3 (2009),
doi:10.1147/JRD.2009.5429057
7. Cloud Computing Use Cases Group: Cloud computing use cases white paper (2010),
http://opencloudmanifesto.org/Cloud_Computing_Use_Cases_Whitepaper-4_0.pdf
8. Distributed Management Task Force: Architecture for managing clouds (2010),
http://dmtf.org/sites/default/files/standards/documents/DSP-IS0102_1.0.0.pdf
9. Distributed Management Task Force: Use cases and interactions for managing clouds (2010), http://www.dmtf.org/sites/default/files/standards/documents/DSP-IS0103_1.0.0.pdf

10. Emeakaroha, V.C., Netto, M.A.S., Calheiros, R.N., Brandic, I., Buyya, R., De Rose, C.A.F.: Towards autonomic detection of SLA violations in cloud infrastructures. *Future Gener. Comput. Syst.* 28(7), 1017–1029 (2012),
<http://dx.doi.org/10.1016/j.future.2011.08.018>, doi:10.1016/j.future.2011.08.018
11. Fortiș, T.F., Munteanu, V.I., Negru, V.: Steps towards cloud governance. A survey. In: *Proceedings of the ITI 2012 34th International Conference on Information Technology Interfaces (ITI)*, pp. 29–34 (2012), doi:10.2498/iti.2012.0374
12. Fortiș, T.F., Munteanu, V.I., Negru, V.: Towards a service friendly cloud ecosystem. In: *Proceeding of the 11th International Symposium on Parallel and Distributed Computing* (2012)
13. Gutierrez-Garcia, J., Sim, K.M.: Self-organizing agents for service composition in cloud computing. In: *2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 59–66 (2010), doi:10.1109/CloudCom.2010.10
14. Kohlborn, T., Korthaus, A., Rosemann, M.: Business and software service lifecycle management. In: *IEEE International Enterprise Distributed Object Computing Conference*, pp. 87–96 (2009),
[doi: http://doi.ieee.org/10.1109/EDOC.2009.20](http://doi.ieee.org/10.1109/EDOC.2009.20)
15. Maule, R.W., Lewis, W.C.: Service evolution lifecycle for service oriented architecture. *IEEE Congress on Services*, 461–462 (2009),
[doi: http://doi.ieee.org/10.1109/SERVICES-I.2009.69](http://doi.ieee.org/10.1109/SERVICES-I.2009.69)
16. Munteanu, V.I., Fortiș, T.F., Negru, V.: An event driven multi-agent architecture for enabling cloud governance. In: *Proceedings of the 2012 Fifth IEEE International Conference on Utility and Cloud Computing* (2012)
17. Munteanu, V.I., Fortiș, T.F., Negru, V.: Service lifecycle in the cloud environment. In: *International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, SYNASC (2012)
18. Naick, I.: Make autonomic computing a reality with IBM Tivoli (2004),
<http://www.ibm.com/developerworks/library/ac-itito/index.html>
19. NIST: Cloud architecture reference models: A survey (2011),
http://collaborate.nist.gov/twiki-cloud-computing/pub/CloudComputing/Meeting4AReferenceArchitecture013111/NIST_CCRATWG_004v2_ExistentReferenceModels_01182011.pdf
20. Oliver, D.: What is SOA governance? (2007),
<http://geekswithblogs.net/SabotShell/archive/2007/02/04/105428.aspx>
21. Stantchev, V., Malek, M.: Addressing dependability throughout the SOA life cycle. *IEEE Transactions on Services Computing* 4, 85–95 (2011),
[doi: http://doi.ieee.org/10.1109/TSC.2010.15](http://doi.ieee.org/10.1109/TSC.2010.15)
22. Talia, D.: Clouds meet agents: Toward intelligent cloud services. *IEEE Internet Computing* 16(2), 78–81 (2012), doi:10.1109/MIC.2012.28
23. Tung, T.: Defining a cloud reference model. In: *International Symposium on Cluster Computing and the Grid*, pp. 598–603. IEEE Computer Society (2011),
[doi: http://doi.ieee.org/10.1109/CCGrid.2011.66](http://doi.ieee.org/10.1109/CCGrid.2011.66)

24. Venticinque, S., Aversa, R., Di Martino, B., Rak, M., Petcu, D.: A cloud agency for SLA negotiation and management. In: Guaraccino, M.R., et al. (eds.) Euro-Par-Workshop 2010. LNCS, vol. 6586, pp. 587–594. Springer, Heidelberg (2011),
<http://dl.acm.org/citation.cfm?id=2031978.2032058>
25. Venticinque, S., Aversa, R., Di Martino, B., Petcu, D.: Agent based cloud provisioning and management - design and prototypal implementation. In: CLOSER 2010, pp. 184–191 (2011)
26. Venticinque, S., Negru, V., Munteanu, V.I., Sandru, C., Aversa, R., Rak, M.: Negotiation policies for provisioning of cloud resources. In: Proceedings of the 4th International Conference on Agents and Artificial Intelligence, pp. 347–350. SciTePress (2012), doi:10.5220/0003747003470350

Exploiting Cloud Technologies and Context Information for Recommending Touristic Paths

Flora Amato, Antonino Mazzeo, Vincenzo Moscato, and Antonio Picariello

Abstract. In the developing of applications for touristic paths planning, context-aware recommendation services are gaining more and more relevance. Recommender applications can accommodate location's dependent information with user's needs in a mobile environment, related to the touristic domain.

In this paper, we propose a touristic context-aware recommendation system based on both the experience of previous users and on personal preferences of tourists. The information are gathered by several heterogeneous sources (sensors, web portals, repositories related to touristic events and locations) and are stored and analyzed in a cloud architecture that is particularly suitable to process and manage the huge amount of data extracted.

Keywords: Recommendation Systems, Knowledge Management, Cloud Computing.

1 Introduction

Travel and tourism industry is one of the most important and dynamic sectors, especially in B2C e-commerce. The tourism industry is regarded as one of the biggest sectors in the world generating an estimated 11% of the global gross domestic product and employing 200 million people and serving 700 million tourists worldwide, a figure which is expected to double by the year 2020.

The great advances of the information technologies are continuously changing the way in which touristic services are performed, enabling people to access accurate and “on-line” information as well as to undertake reservations and plans in real time, thus reducing costs and satisfactions w.r.t. conventional methods.

Flora Amato · Antonino Mazzeo · Vincenzo Moscato · Antonio Picariello
Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione,
University of Naples “Federico II”, via Claudio 21, Naples, Italy
e-mail: {flora.amato,mazzeo,vmoscato,picus}@unina.it

Just to make an example, taking a quick look at what is happening in the last few years, people search for information before traveling, usually make online air-ticket bookings, and hotel and room reservations, decide the kind of restaurants to go and the events to attend. As a matter of fact, itinerary planning is often a difficult and time consuming task for people, especially if they visit a destination for the first time: it involves substantial research to identify what are the main locations to visit, the time spending at each location, what should be next locations, and the time it will take to get from one place to another and of course how to reach the next places (public transportation, taxi, and so on). Without any prior knowledge, at the moment a lot of people still rely on travel books or on recommendation travel blogs.

However, these options have a lot of problems: travel books do not cover all cities/locations and, perhaps more importantly, are not free; travel blogs reflect a single person's view, with no guarantees that the information provided are reliable. In this framework, recommender systems are becoming more and more important as applications that applications exploit to suggest products and provide people with useful information to facilitate their decision-making processes. In other words, such kind of system implicitly assumes that it can map user needs and constraints, through appropriate recommendation algorithms, and convert them into a number of location-events-products selections using appropriate knowledge that is retrieved and analyzed by the system[4, 5, 8].

In this paper we propose a touristic context-aware recommendation system based on both the experience of previous users and on personal preferences of tourists and dependent on the *context*. The information are gathered by several heterogeneous sources (sensors, web portals, repositories related to touristic events and locations). All the information are stored and analyzed in a cloud architecture that is particularly suitable to process and manage the huge amount of data extracted.

2 Related Works

The focus of the paper is the introduction of context-aware recommendation services for touristic paths planning: in this framework, *recommendation* is the problem of estimating *ratings* - sometimes called also *utilities* - for the set of items that has not yet been seen by a given user [10]. In *Content Based recommender systems*, the rating r_j^i of item o_j is estimated using the utilities $r(u_i, o_k)$ assigned by the user u_i to items o_k that are in some way “similar” to item o_j . One of the main drawbacks of these techniques is that the system can only recommend items that are similar to those already rated by the user itself (*overspecialization*).

Collaborative Filtering is, in the opposite, the process of filtering or evaluating items using the opinions of other people. Collaborative systems predict the rank of items r_j^i for a particular user u_i based on the utility $r(u_h, o_k)$ of items o_k previously rated by other users u_h “similar” to u_i . It takes its root from something human beings have been doing for centuries: sharing opinions with others. Content-based filtering and collaborative filtering are usually combined by means of the *hybrid approach* that helps to avoid certain limitations of each method.

In the area of recommendation systems, in the last few years, the use of additional contextual information has recently brought to the introduction of the *Context-aware Recommender Systems (CARS)*[10]. In the *Contextual Pre-filtering* techniques context information are used to initially select the set of relevant items, while a classic recommender is used to predict ratings. In the *Contextual Post-filtering* approaches context are used in the last step of the recommending process to “contextualize”, for each user, the output of a traditional recommender.

3 Recommending Personalized Touristic Paths

The problem of building effective context-aware recommendation services, able to support an intelligent planning of *touristic paths*, implies the identification of “items” that are most likely to satisfy the interests of a user at any given point of his/her exploration depending on the context conditions. Here we need to address some fundamental research questions: (i) how can we model the context? (ii) how can we select a set of objects that are good candidates for a recommendation and how can we rank the set of candidates? (iii) how can we organize the recommended objects in visiting paths? In other words, which kind of *recommending strategy* can we adopt?

In our idea, each interesting item (hotel, restaurant, museum, etc.) that will compose a touristic path is characterized by a set of *metadata*, corresponding to specific values of taxonomic attributes of an available a-priori knowledge useful for touristic applications, and by a set of *context information*, describing the “situation” for the place captured by apposite sensors. The context is represented by means of the well-known *key-value model*[1] that uses some pre-defined variables to describe several conditions of a visiting place. We define four main classes of context parameters (dimensions of the context): (i) *time* dimension of the considered place (the relative time requested to user to reach the place, the opening/closing time, etc.); (ii) *location* dimension for the considered place (the address, the actual position in terms of GPS coordinates, the relative position of user respect to the place, etc.); (iii) *environmental* dimension for the considered place in terms of current weather and environmental conditions (e.g. temperature, humidity, rainfall degree, wind, season, moment of the day, etc.); (iv) *social* dimension for the considered place in terms of number of users (e.g. visitors, guests, etc.) close to the considered place and the number of positive/negative feedbacks.

In addition to the context data, we use a *Knowledge Base* containing a set of rules able to provide, in each moment and for each recommended, item a *comfort degree* on the base of some context parameters values.

For example, the following part of rule *very_low_degree* \leftarrow (*env_cond.temp* \geq 30° \wedge *env_cond.humidity* \geq 80%) indicates a very low comfort degree for an outdoor archeological site. The comfort degree is eventually used in the post-filtering recommendation activity to arrange the order of items of the same type (if there is more than one alternative in the path). The context parameters values can be

periodically obtained using apposite web services or exploiting the sensors available on the place.

Once captured the context status of a place, the basic idea is that when a user is interested in the suggestion of a touristic path, the system: (i) receives the request and collects the information about user preferences/needs, for example the list of interesting items (e.g. museums, restaurants, hotel) and of the related constraints (e.g. he/she would like to visit museums with baroque pictures, then to eat in a cheap restaurant offering pizza, and finally to accommodate in a comfortable hotel near the sea); (iii) selects a set of candidate objects for each type of objects that satisfy the user needs (pre-filtering strategy); (iv) ranks these objects using a proper recommendation strategy; (v) arranges such objects in apposite touristic visiting paths considering the comfort degree and the available cartography (post-filtering strategy).

We use as recommendation strategy an *importance ranking method* that some of the authors have proposed in [2]. Such a method combines metadata information of objects (items), past behavior of individual users and overall behavior of the whole community of users and context information. Our basic idea is to assume that when an object o_i is chosen after an object o_j in the same browsing session, this event means that o_j “is voting” for o_i . Thus, our idea is to model a browsing system for a set of objects O as a labeled graph (G, l) , where $G = (O, E)$ is a directed graph and $l: E \rightarrow N$ is a function that associates each edge in $E \subseteq O \times O$ with a particular weight, representing the number of times that an object o_i was accessed immediately after an object o_j . As a consequence, a link from o_j to o_i indicates that part of the importance of o_j is transferred to o_i .

Given a labeled graph (G, l) , we can formulate the definition of *preference grade* of an object for a user as $\rho(o_i) = \sum_{o_j \in PG(o_i)} w^i_j o_j$, where $PG = \{o_j \in O | (o_j, o_i) \in E\}$ is the set of predecessors of o_i in G , and w^i_j is the normalized weight of the edge from o_j to o_i . For each $o_j \in O$, $\sum_{o_i \in SG(o_j)} w^j_i = 1$ must hold, where $SG = \{o_i \in O | (o_j, o_i) \in E\}$ is the set of successors of o_j in G . It is easy to see that the vector $R = [\rho(o_1) \dots \rho(o_n)]^T$ can be computed as the solution to the equation $R = C \cdot R$, where $C = \{w^i_j\}$ is an ad-hoc matrix that defines how the importance of each object is transferred to other objects and can be seen as a linear combination of *browsing matrices* [3]. Once obtained the discussed matrices, our main goal is to compute customized rankings for each individual user considering his/her preferences and the context information. In this case, we can then rewrite previous equation considering the ranking for each user as $R_l = C \cdot R_l$, where $R_l = [\rho(o_1) \dots \rho(o_n)]^T$ is the vector of preference grades, customized for a user u_l .

C , under certain assumptions and transformations, is a real square matrix having positive elements, with a unique largest real eigenvalue and the corresponding eigenvector has strictly positive components, and the discussed equation can be solved using the *Power Method* algorithm. It is important to note that C takes into account the user’s preferences and the context and does not have to be computed for all the database objects, but it needs to be computed only for those objects that are

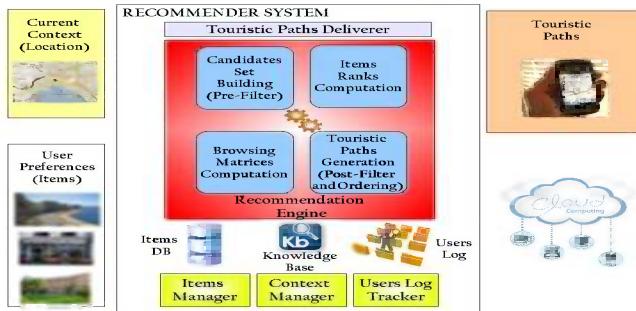


Fig. 1 System Overview

good candidates, i.e. the objects that are closest to users and in which a user is really interested in (pre-filtering strategy). To compute the set of candidates that are more similar to user preferences, we model a single user as a particular set of objects and consider only the browsing patterns containing a sequence of objects that are “similar” to the user objects and that have been visited by any user in the same order. In particular, we exploit a semantic similarity[6, 7] based on the object metadata that has been computed used the Li-Bandar-McLean metric for semantic relatedness of concepts based on a vocabulary [9]).

Finally, the list of suggested items is organized in apposite visiting paths: they are not fixed and are arranged on the base of environmental situations and comfort degrees (post-filtering strategy).

4 The System Overview

Figure 1 shows an overview of our recommender system, which takes as input the current context in terms of user location and preferences (items of interest) and generates a touristic path. We can distinguish the following main components. *Items Manager* - It is a repository manager that stores the items to be suggested with the related descriptions. *Users Log Tracker* - It is a module devoted to capture and store - in an appropriate format - all the users’ browsing sessions in terms of accessed items during their explorations. *Context Manager* - It is a module devoted to gather from sensors, web portals and other kind of repository the context information. *Recommendation Engine* - It is the system core that for each user and on the base of current context dynamically proposes a set of recommended objects ordered on the base of their utility; in particular, it is composed by: (i) a *Browsing Matrices Computation* Module - able to transforms the collected browsing sessions into two matrices: a global matrix which takes into account the overall browsing behavior of the users, and a local matrix which considers the behavior of a single user; (ii) a *Candidate Set Building* Module - computes the subset of items that fit with users needs; (iii) a *Items Ranks Computation* Module - performs the ranking of the

selected candidates for recommendation; (iv) a *Touristic Paths Generation Module* - capable of arranging the suggested items on the base of environmental situations and the comfort degrees. *Items Deliverer* - It aims at delivering recommended paths to each user in a format that will depend on the user profile and device. All the services are provided by means of a dedicated Cloud Computing infrastructure that allows to deal with big collections of data and numerous user accesses, ensuring high performances, scalability and security characteristics for the whole system [12],[11].

5 Conclusions

Our proposal represents an extension of a recommender system supporting touristic paths planning. We have shown that the customized recommendations may be computed combining several features of objects, past behavior of individual users and overall behavior of the entire community of users and context information.

References

1. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)* 23(1), 103–145 (2005)
2. Albanese, M., d’Acierno, A., Moscato, V., Persia, F., Picariello, A.: Modeling recommendation as a social choice problem. In: *Proceedings of the fourth ACM Conference on Recommender Systems*, New York, NY, USA, pp. 329–332
3. Albanese, M., d’Acierno, A., Moscato, V., Persia, F., Picariello, A.: A multimedia semantic recommender system for cultural heritage applications. In: *2011 Fifth IEEE International Conference on Semantic Computing (ICSC)*, pp. 403–410. IEEE (2011)
4. Amato, A., Di Martino, B., Venticinque, S.: A semantic framework for delivery of context-aware ubiquitous services in pervasive environments. In: *2012 4th Int. Conf. on Intelligent Networking and Collaborative Systems (INCoS)*, pp. 412–419 (2012)
5. Amato, A., Di Martino, B., Venticinque, S.: Semantically augmented exploitation of pervasive environments by intelligent agents. In: *2012 IEEE 10th Int. Symp. on Parallel and Distributed Processing with Applications (ISPA)*, pp. 807–814 (2012)
6. Amato, F., Mazzeo, A., Moscato, V., Picariello, A.: Semantic management of multimedia documents for e-government activity. In: *Int. Conf. on Complex, Intelligent and Software Intensive Systems, CISIS 2009*, pp. 1193–1198. IEEE (2009)
7. Amato, F., Mazzeo, A., Moscato, V., Picariello, A.: A system for semantic retrieval and long-term preservation of multimedia documents in the e-government domain. *International Journal of Web and Grid Services* 5(4), 323–338 (2009)
8. Amato, F., Mazzeo, A., Penta, A., Picariello, A.: Knowledge representation and management for e-government documents. In: Mazzeo, A., Bellini, R., Motta, G. (eds.) *E-Government Ict Professionalism and Competences Service Science*. IFIP, vol. 280, pp. 31–40. Springer, Boston (2010)
9. Budanitsky, A., Hirst, G.: Semantic distance in wordnet: An experimental, application oriented evaluation of five measures. In: *Proceedings of the Workshop on WordNet and other Lexical Resources* (2001)
10. Kantor, P.B., Ricci, F., Rokach, L., Shapira, B.: *Recommender systems handbook*. Recherche 67, 02 (2010)

11. Moscato, F., Aversa, R., Di Martino, B.: An analysis of mosaic ontology for cloud resources annotation. In: 2011 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 973–980. IEEE (2011)
12. Moscato, F., Aversa, R., Di Martino, B.: An ontology for the cloud in mosaic. In: Cloud Computing: Methodology, System, and Applications (2011)

An FPGA-Based Smart Classifier for Decision Support Systems

Flora Amato, Mario Barbareschi, Valentina Casola, and Antonino Mazzeo

Abstract. In recent years, the accuracy and performance of decision support systems have become a bottleneck in many monitoring applications. As for the accuracy, different classification algorithms are available but the overall performance are related to the specific software implementation. In this paper we propose a novel hardware implementation to fasten a decision tree classifier. We also present the evaluation of our architecture by putting in evidence the positive performance results obtained with the proposed implementation.

1 Introduction

In modern decision support systems there is the need to improve the performance in terms of detection, reliability and real time capabilities. These features are usually in contrast among each other. Furthermore, many monitoring systems rely on heterogeneous data acquisition tools (sensors, video, historical and simulated data,...) and on data elaboration and correlation to prune non significant information [8, 11]; this heterogeneity, even if desirable, introduces the need to further interpret what data really represents to reduce false alarms and detect even weak alarm conditions. The adoption of semantic techniques and inference knowledge is a must to cope with heterogeneity, even in large environment as in the clouds, nevertheless at the same time real-time capabilities must be ensured.

In some recent works [7, 8] we proposed an innovative approach for smart event detection and semantically enriched phenomena comprehension: the knowledge base is inferred off line for further comprehension and model prediction refinement, while a light classifier is used to quickly raise an alarm. In many monitoring applications, specific classifiers are adopted to elaborate huge amount of data and they

Flora Amato · Mario Barbareschi · Valentina Casola · Antonino Mazzeo
Department of Electrical Engineering and Information Technology
University of Naples Federico II, Napoli, Italy
e-mail: {flora.amato,mario.barbareschi,casolav,mazzeo}@unina.it

need appropriate computational resources. A classification system could adopt hardware solutions to boost up performances but, the variable nature of classifiers (they need to periodically change their parameters when too false positives are counted or when some errors are caused by misconfigurations) does not match hardware characteristics. In fact, hardware cannot change its functionality, the only way to change hardware behaviour is to re-project the whole hardware. Indeed, approaches based on Field Programmable Gate Array (FPGA) technology, that allows to easily reconfigure hardware, can been used to implement classifiers, too. In some recent works [9] FPGA architectures were used for implementing algorithms for face detection, for image recognition and videosurveillance applications [6] or for real time object recognition. Indeed, a classifier is composed by a learner and by a predictor. The learner does not change its functionalities during application lifetime. It has to process the training set at start-up and the result of its computation defines how the predictor has to work. On the other hand, the predictor has to quickly evaluate and classify new data according the configuration parameters. In this paper we introduce an optimized hardware implementation based on reconfigurable hardware to implement a decision tree classifier. We also propose an application that automatically produces the VHDL code for FPGA synthesis, optimized for the chosen classifier. First experimental results are very promising, they report a response time of nanoseconds order. The reminder of the paper is structured as follows, in Section 2 the general architecture of a smart classifier is presented. In Section 3 some details of the adopted classifier is presented. In Sections 4 and 5 we illustrate in details the proposed hardware implementation to optimize the tuning and configuration of the Predictor component and some evaluation results will be discussed. Finally, in section 6 some conclusion and future work will be discussed.

2 A Two Steps Decision Support System

In previous works we proposed a smart Decision Support Systems (DSS) able to perform a quick classification of potential alarms and to refine the final decision based on semantic inferences on data and events [1, 2]. The system core was based on two different components:

1. a Smart Event Classifier;
2. a Post Reasoner.

In real environments, a complex potential hazard cannot be easily detected by using data from a single device, nevertheless sensor networks and data heterogeneity do not help in correlating data. So, the main idea behind that proposal was related to semantically enrich sensor data gathered by different sensors and correlate them for “operator-aware” event detection.

We developed a complex architecture made of different modules to: (i) semantically enrich data, (ii) implement the smart real-time classifier, (iii) implement the post-reasoner. In Figure 1, the DSS main modules are reported. We suppose that the sensor data source consists of heterogeneous sensor networks (WSN, camera,

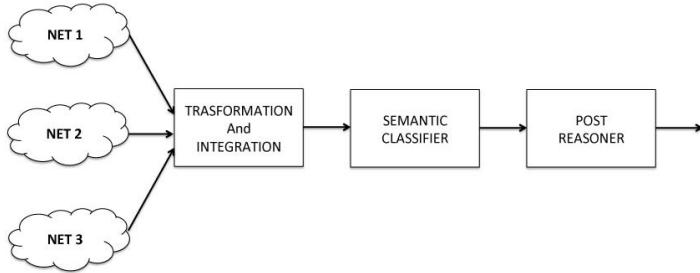


Fig. 1 Decision Support System architecture

intrusion device, etc) measuring different parameters. Data are gathered from sensor nodes and can be accessed through a specific sensor gateway. Gateway sensors code data in XML.

The system is made of three modules:

The **Transformation and integration module** gathers and integrates data coming from heterogeneous sensors. Data are here semantically enriched to add information and description about measures and sensors [1] and automatically encoded in RDF files. They contain unique references to descriptions of the measured variables and parameters of each network and are then integrated into a single RDF file, containing instances of the measured values that will populate the ontology.

The **Semantic Classifier module** implements a rule-based classifier: the combination of measurements allows classifying events, if there is a particular event, classified as critical, the system raises the corresponding alarm. The smart classifier operates on semantically enriched data, so, a rule can be expressed by combining atomic events from heterogeneous sources [3]. We chose a well-known classifier [12] to build the predictive model, it is a decision tree classifier made of a learner and a predictor component. The learner module uses a training set data in order to tune the predictor parameters, while the predictor is responsible to classify data and decide if alarm conditions occur. Furthermore, on a periodic basis or even when mis-classified events occur, the learner can recalculate the parameters of the predictor on the basis of a new training set, properly built to refine the behaviour of the classifier.

The overall performance, in terms of response time and accuracy of the whole decision system, rely on the performance and accuracy of this module and its design is very crucial. For this reason, in this paper we propose to synthesize it with a reconfigurable hardware as an FPGA. The expected advantages are twofold:

1. we expect to boost up the performance [13] and meet real time constraints,
2. we can easily reconfigure the whole predictor in case of parameter changing.

In the next Sections, we will illustrate in details this component and its hardware implementation.

The **Post Reasoner module** allows to analyze causes and reasons of the critical events that have taken place (basic function for an operator) and to refine the

procedures and rules of classification if a false alarm rate is not acceptable. Reasoning operations were performed on data in order to extract inferred knowledge recurring to a Pellet reasoner. The real time classification actions feed the knowledge base for the Post Reasoner component. The just built ontology is stored in a repository (Triple Stores) and can be used off-line through the adoption of semantic query languages as SPARQL. Through queries, the post reasoner is able to understand and explain to end users the meaning of the alarms and their causes. Details about this component are out of the scope of this paper and can be found in some previous works [7].

3 The Fast Classification Algorithm

In order to semantic classify heterogeneous sensed data, we have implemented, among the different classifiers available in literature, the decision tree classifier proposed by [12]. The well-known classifier is composed by a learner module for building the predictive model and a predictor component for performing decision activity on the data. In binary decision mechanisms and the set of decision rules are modelled as a tree where nodes represent classes associated to atomic events to be detected, and branches represent conjunctions of conditions; i.e. conditions on the sensed data that lead to composed event classes. In order to define the branch rules of the decision tree, a domain expert manually defines a training set, made of already classified data. The predictor is a parametric system: parameters, given by the learner module, determine the classification function. Periodically, or when mis-classified events happen, the learner can recalculate the parameters on the basis of a new training set in order to refine the behavior of the classifier. Indeed decision trees are chosen with the aim of easing further refinement [10]. In fact, it is a white box model and domain operators will be able to easily interpret decision tree results after a brief explanation. This kind of model allows to quickly identify if specific conditions of mis-classification occur and, consequently, to re-tune the model. Moreover if a given situation is observed in the model, it is easy to explain the conditions by recurring to the boolean logic. The Classifier takes in input the semantically enriched data codified in RDF. By using an XLST engine these data are reported in a table containing a row for each sample and a column for each relevant measured parameter. Element i, j , then, contains the measure value of the sample i for the parameter j . In the remainder of this section, we are going to give some details on the classifier algorithm and how to use it. These details will be useful to understand the different optimization that can be performed at hardware level.

The algorithm describes the fundamental steps for building the model, it works recursively on data taken from the training set. At each step the data set is split, based on a condition of a chosen feature (defined by thresholds). The selection of the feature is performed on the basis of an *Entropy Test*[4].

If T is the set of samples, and $freq(C_j, T)$ is for the number of samples in T that belong to class C_j . Let $|T|$ be the number of samples in the set T . Then the entropy of the set T is defined as:

```

Data: Training Set  $T$ , Output Class  $C = C_1, \dots, C_k$ 
Result: Prediction Model outputted by learner
/* Base Case */ 
if  $T \neq \emptyset$  then
  Create a single leaf in which all the sample having label  $C_j$  /* The decision
  tree for  $T$  is a leaf identifying class  $C_j$  */ 
end
if  $T = \emptyset$  then
  Creates a single leaf that has the label of the most frequent class of the samples
  contained in the parent node /* heuristic step leading to
  misclassification rates */ 
end
/* Recursive Case */
if  $T$  contains samples that belong to several classes then
  foreach Feature  $f$  do
    Find the normalized information gain by splitting on  $f$ , based on an Entropy
    Test Value
  end
  Let  $f_i$  be the attribute with the highest normalized information gain; Create a
  decision node that splits on  $f_i$ ; /* node associated with the test
  */
  Create a branch for each possible outcome of the test; Execute the algorithm on
  each branch (corresponding to a subset of sample) /* partitioning of
   $T$  in more subsets according to the test on the
  attribute obtained by splitting on  $f_i$  */ 
end

```

Algorithm 1. Description of the algorithm for building a decision tree model

$$Info(S) = - \sum ((freq(C_j, T)/T) \cdot log_2(freq(C_j, T)/T))$$

The algorithm computes the value of $Info$ for all T_i partitioned in accordance with n conditions on a feature f_i :

$$Info_{f_i(T)} = \sum_i ((T_i/T) \cdot Info(T_i))$$

The features that maximizes the following Gain value are selected for the splitting:

$$Gain(f_i) = Info(T) - Info_{f_i(T)}$$

Running this algorithm, we built a tree data structure; evaluating the conditions requires the visiting of all paths in the tree. In the predictor module, even in the best case of a balanced tree, the complexity of visiting a tree path, in order to evaluate all the conditions leading to a leaf, is $O(\log_2(n))$, with n the number of tree nodes. Consequently, implementing the predictor in software implies that the node conditions of a given tree path are evaluated sequentially. We propose a reconfigurable hardware architecture to reduce the time spent in the decision tree path discovery, by elaborating the whole algorithm in only 2 sequences. Hardware implementation, intrinsically concurrent, of the predictor allows us to elaborate the conditions of the

nodes of the tree branches in a concurrent way. Furthermore hardware implementation allow us to set real time constraints, too. In fact, during the hardware design phase, we can estimate the delay associated with the critical path and we can build the real time conditions on the basis of such values. In the next sections we will show that our hardware approach increase the throughput of 10^3 times, providing results of the order of $\approx 10ns$.

4 An Optimized Hardware Implementation of the Decision Tree Predictor

To get an hardware description of the predictor, first of all we needed the training set. So we collected some meaningful data and we manually labelled them with a classification value. The training set is the input of an automated process which produced the required hardware.

In this process we can identify two phases. The first one aims at defining the learner, that computes the predictor parameters. We choose KNIME [5], a data mining framework, to complete the first phase. With the decision tree blocks of KNIME, we are able to retrieve the predictor parameters. Generally this phase is done off-line, periodically or when errors, caused by a misconfiguration, occur. For this reason there are not strict time constraints in this phase. The second phase aims at building and using the predictor. Due to real time constraints, we will implement it in hardware. In this paper we proposed an application that automatically produces the VHDL code for FPGA synthesis, optimized for the chosen classifier.

4.1 Predictor Phase

As previously said, a decision tree predictor is typically implemented as a visiting algorithm, that is an exploration of the tree, from the root to leaves, node by node in a sequential way. In fact the algorithm has to evaluate each decision to determine in which branch it has to continue until it reaches a leaf. We propose a hardware architecture to dramatically reduce the time spent in the decision tree path discovery, computing the whole algorithm in only 2 sequences. Thanks to the hardware features, we implemented the visiting algorithm with a different approach:

1. We simultaneously evaluate all predicates contained in all tree nodes;
2. We automatically define a boolean function that implements the visit and verify which leaves are effectively reached according to the classified class.

All the decisions are not computed in sequence, but in parallel. Each decision is a boolean value: for this reason the visiting algorithm can be treated as a multi-output boolean function, in which the inputs are the decisions and the outputs the classes. Also this boolean function could be optimized and reduced in order to increase the performance. It can be implemented as a sum of products (SOP); in particular a path that leads to a given class C_i is implemented as an *anding* function of the boolean

decisions along the path. All *and*ing functions associated to different paths that lead to the same class C_i are in *oring*:

$$C_i = \bigvee_{P \in \text{Path set of } C_i} \left(\bigwedge_{\text{Decision Node} \in P} \text{decision} \right)$$

We can characterize a single class function by the *Class Characteristic*, that is the maximum path length plus the number of paths that lead to the class. As shown in Figure 2, our architecture computes all the decisions (rounded rectangles) in parallel, and each result is given in input to the boolean net, that decides which class the input belongs. In particular, to compute all decisions very quickly, we have implemented in VHDL a generic *Decision Box*. It computes a predicate between two values: $D = A \rho C$. The effective input of the *Decision Box* is only the first operand (A) representing the new data, while the operation (ρ) and the second operand (C) are pre-defined, decided during the learning phase. The output D is a boolean variable.

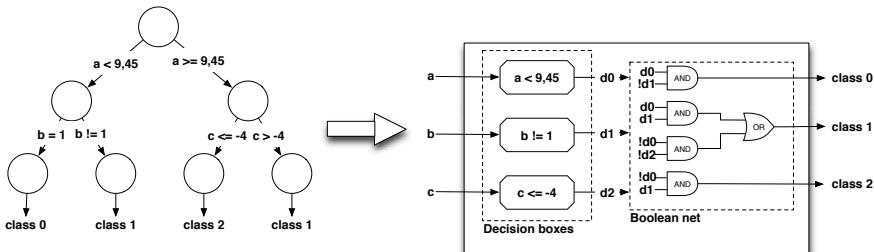


Fig. 2 Predictor hardware architecture

The *Decision Box* has 3×6 different configurations, as it supports 3 different data types (single precision floating point, 32 bit signed integer, 32 bit unsigned integer) and 6 different comparing operations ($<$, \leq , $=$, \neq , \geq , $>$).

The *Decision Boxes* outputs, the decisions, have to be analyzed by the second part of the architecture to classify the input data. In our architecture this task is executed by the *Boolean Net*.

These 2 components are automatically produced by the PMML2VHDL tool in 2 VHDL files at the end of the Learning phase, described in the next section.

4.2 Learning Phase

In the Learning phase, we used the KNIME tool that supports us for the entire analysis process and to realize the learner, accordingly to the algorithm previously described. It offers data access, data transformation, initial investigation, powerful predictive analytics, visualization and reporting in an integrated graphical interface.

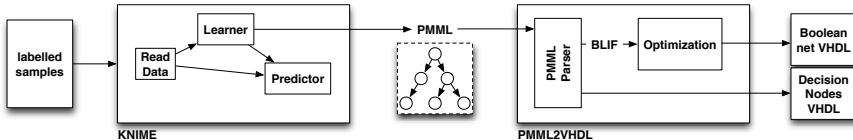


Fig. 3 Learning phase

As illustrated in Figure 3, in the learning phase, KNIME loads the training set file containing data that were manually labelled and previously classified. According to the classification process, KNIME produces in output an XML file, the PMML, that describes the obtained tree.

We developed an application, named PMML2VHDL, that automatically produces the VHDL files describing the Predictor. The Predictor hardware has been optimized for this specific classifier, for this reason we have developed two different components and, consequently, the PMML2VHDL application will produce two different hardware description files: the *Decision Nodes VHDL* and *Boolean Net VHDL*.

The first one contains all predicates that have to be evaluated. The second describes the optimized boolean net for class assignment, so it contains a function for each class defined into the decision tree predictor. To produce an optimized boolean net we used Berkeley SIS tool. So PMML2VHDL tool produces an intermediate result of the boolean net, that is the raw net described in BLIF format. With heuristic method, SIS minimizes and optimizes the net, returning another BLIF file. This last file is translated in a VHDL data flow description, ready to be synthesized.

5 Evaluation of the Proposal

In this section we are going to illustrate some experimental results about our proposal. We took about 100 samples from a local sensor network. The sensors collected temperature, humidity, pressure, battery level and GPS position every 10 seconds. We manually labelled data with an alarm value to define the training set and configure the predictor parameters with KNIME. We generated and optimized the VHDL with our application.

We have performed several experiments by changing each time the number of alarm classes and the training set size and we used Xilinx ISE and a Virtex-5 XC5VLX110T to synthesize and test the architecture. We evaluated the response time (the throughput) of our FPGA-based architecture and the area used to implement the components. Note that algorithm precision is not evaluated as it is not affected by this implementation.

The **Decision Box** implementation can have 3×6 different configurations. To obtain a higher throughput we implemented *Decision Box* in pipelined version with depth 2. In Table 1 the main implementation characteristics are reported for the

Table 1 Evaluation of *Floating Point Decision box* with latency 2

Parameter	<	\leq	\leq	$>$	\neq	\geq
Slices LUTs	43	27	44	32	44	44
Slices Flip-Flops	43	28	45	33	45	45
Delay (ns)	1,779					
Throughput (float/s)	562113546,9					

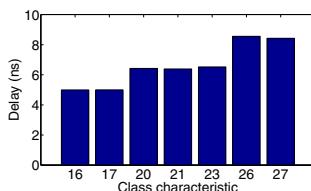
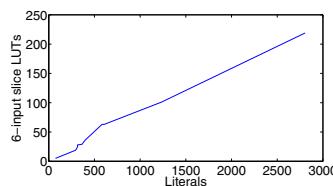
6 different operations; the interesting result is that we get about 560 millions of computed decisions per second, with an used area of about 44 LUTs per box.

The **Boolean Net** delay depends on the predictor tree implementation, as all inputs (the decisions) have the same arrival time. Our hardware implementation computes a classification for a given class as a SOP. For this reason the delay time for an output class depends on the number of paths that leads to the class and on the maximum path length. In fact in the SOP implementation the delay is a function of the higher fan-in of product terms and of number of product terms. In the previous section, we defined the *Class Characteristic* to get a meaningful parameter for comparing the *Boolean Net* delays. As shown in Figure 4, the delay does not linearly increase with the *Class Characteristic* but it has a step behavior. This is primary due to the chosen FPGA, as the gate delay is the same of a LUT if the fan-in is less than or equal 6, so for similar *Class Characteristics* the delay was the same.

It is also possible to note that the delay time is significantly longer than *Decision Boxes* time, about 7ns, but this value is considerably better than a software implementation of the predictor module; indeed, with KNIME [5], we obtained a delay of $\approx 10\text{ms}$.

To avoid bottle-neck effect on the throughput, it's possible to implement a high performance pipelined version, too.

Finally, we considered the *Boolean Net* used area in terms of FPGA LUTs. By using the Virtex5 Family we have 4 LUTs for each slice. As illustrated in Figure 5, results show a linear trend with the growing of function literals: the mean slope is about 11,84 lits/LUTs.

**Fig. 4** Step behavior of *Class Characteristic* and net delay**Fig. 5** Used FPGA LUTs on number of literals

6 Conclusions and Future Work

In recent years, the need for smart monitoring systems has grown and, in particular, the accuracy and performance of decision support system has become a bottleneck. As for the accuracy, different classification algorithms are available but the overall performance is related to the specific software implementation.

In this paper we proposed a process to implement in hardware a reconfigurable decision tree classifier. Furthermore we proposed an innovative architecture to fasten the classification process, it is composed by a set of *Decision Boxes* to compute in parallel all decisions and by a *Boolean Net*, to effectively compute the classification. We evaluated the proposal from different perspectives, putting in evidence the great performance obtained with the hardware implementation. In future work we intend to enhance the proposal by introducing automatic pruning rules in the tree model with the application, in order to improve performance of the predictor *Boolean Net*, that is the most critical component.

References

1. Amato, F., Casola, V., Gaglione, A., Mazzeo, A.: A common data model for sensor network integration. In: Proceedings of the 4th International Conference on Complex, Intelligent and Software Intensive Systems, pp. 1081–1086 (2010)
2. Amato, F., Casola, V., Gaglione, A., Mazzeo, A.: A semantic enriched data model for sensor network interoperability. *Simulation Modelling Practice and Theory* 19(8), 1745–1757 (2011)
3. Amato, F., Casola, V., Mazzeo, A., Romano, S.: A semantic based methodology to classify and protect sensitive data in medical records. In: 2010 Sixth International Conference on Information Assurance and Security (IAS), pp. 240–246. IEEE (2010)
4. Berger, A.L., Pietra, V.J.D., Pietra, S.A.D.: A maximum entropy approach to natural language processing. *Computational linguistics* 22(1), 39–71 (1996)
5. Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Thiel, K., Wiswedel, B.: Knime - the konstanz information miner: version 2.0 and beyond. *SIGKDD Explor. Newsl.* 11(1), 26–31 (2009)
6. Bhowmik, D., Amavasai, B.P., Mulroy, T.J.: Real-time object classification on fpga using moment invariants and kohonen neural networks. In: IEEE SMC UK-RI Chapter Conference 2006 on Advances in Cybernetic Systems, pp. 43–48 (2006)
7. Casola, V., Esposito, M., Mazzocca, N., Flammini, F.: Freight train monitoring: A case-study for the pshield project. In: Proceedings - 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2012, pp. 597–602 (2012)
8. Casola, V., Gaglione, A., Mazzeo, A.: A reference architecture for sensor networks integration and management. In: Trigoni, N., Markham, A., Nawaz, S. (eds.) *GSN 2009. LNCS*, vol. 5659, pp. 158–168. Springer, Heidelberg (2009)
9. Cho, J., Mirzaei, S., Oberg, J., Kastner, R.: Fpga-based face detection system using haar classifiers. In: *FPGA*, pp. 103–112 (2009)
10. Deng, H., Runger, G.C., Tuv, E.: Bias of importance measures for multi-valued attributes and solutions. In: Proceedings of the ICANN, pp. 293–300 (2011)
11. Ficco, M.: Security event correlation approach for cloud computing. *Journal of High Performance Computing and Networking* 7(3) (2013)

12. Liu, B., Ma, Y., Wong, C.K.: Improving an association rule based classifier. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 504–509. Springer, Heidelberg (2000)
13. Wittig, R.D., Chow, P.: Onechip: An fpga processor with reconfigurable logic. In: IEEE Symposium on FPGAs for Custom Computing Machines, pp. 126–135. IEEE (1996)

Agents Modeling under Fairness Assumption in Event-B

Irina Mocanu, Lorina Negreanu, and Adina Magda Florea

Abstract. Multi-agent systems, which are composed of autonomous agents are present in applications that span a wide range of domains: ambient intelligence, telecommunications, finance, Internet, energy, health. Therefore, it is critical to have rigorous, effective design and verification methods to ensure their development. In this paper, we present a formal modeling and proof of a multi-agent system for requesting services, under the fairness assumption. The model is specified and verified using Event-B and the Rodin platform.

1 Introduction

This paper specifies and verifies, using the Event-B [5], [6] formal specification method, a multi-agent system for requesting services under the fairness assumption. This should be read as an extension of the model we previously specified in [9], who will be integrated in an ambient intelligent system.

Fairness plays an important role in software specification, verification and development. Fairness properties state that if something is enabled sufficiently often, then it must eventually happen [8]. This becomes important in our model when the requests are satisfied. It is possible that a request cannot be satisfied for an indefinite period of time while other requests continue normally. This may occur if the satisfying scheme of the requests is unfair.

The paper is organized as follows: Section 2 describes the proposed system, Section 3 presents the refinement of the model specified in [9] while fairness constraints are stated using the Event-B method, Section 4 lists conclusions and future work.

Irina Mocanu · Lorina Negreanu · Adina Magda Florea
University “Politehnica” of Bucharest,
Computer Science Department,
Splaiul Independentei 313,
060042 Bucharest, Romania
e-mail: {irina.mocanu, lorina.negreanu, adina.florea}@cs.pub.ro

2 System Description

We refine the former specification of the system for managing requests for services [9] under the strong fairness assumption [8]. Consider a supervised person that wants to do an activity at a relaxing centre (e.g. swimming, physiotherapy or talking to somebody). Since the person's well being is important we need to have a system that will manage his request based on some measured parameters.

The system we propose is composed of several agents [9]: (i) an agent to interrogate the ambient factors - *Temperature Agent*; (ii) an agent to verify the health status of the supervised person -*Pulse Agent*; (iii) an *User Agent* associated with the user; (iv) *Service Agents* (there are n agents - each agent has a specialization and a maximum available time) and (v) a *Community Agent* (this agent knows all the services available and unavailable). The user makes a request and the *User Agent* will analyze the request by computing the priority and the duration of the service using the monitoring information (health status of the person and performed activities). The *User Agent* will send a verification message to the *Temperature Agent* and to the *Pulse Agent* in order to verify if the supervised person can perform that activity. If these parameters (the ambiental temperature/pulse) are in normal values for that person, the *User Agent* will send a request message to the *Community Agent* (*new_request*). If the answer for the request is not received in a predefined time, the *User Agent* will send a message to the *Community Agent* (*modify_request*) in order to increase the priority of the requested service for that user. The request will be canceled by the *User Agent* (*cancel_request*). If there is available time for requested service, the corresponding *Service Agent* will inform the *User Agent* through the *Community Agent* (*satisfy_request*). Some services may be requested more frequently than others. Thus the *Community Agent* will request to increase the maximum available duration for a *Service Agent*, by taking some available time from other services (*request_available*). The *Service Agent* associated with the requested service with fewer requests will add some extra time to that service (*release_available*).

3 Formal Specification

In the previous work [9] we specified the request as the main modeling element. A request is defined as a member of the set of requests. It has a *status* (*pending* or *satisfied*), a *service* and a *duration* associated, that can be accessed through the functions

$$status \in requests \rightarrow STATUS,$$

where *status* assigns a status to each request,

$$reference \in requests \rightarrow SERVICES,$$

where *reference* assigns a service to each request,

$$duration \in requests \rightarrow N^*,$$

where *duration* assigns a duration to each *request* (with the assumption that if a service is requested, the duration is at least 1). The status of the request is changed

into *satisfied*, if the duration of the requested service is either less than or equal to the time availability of the service, defined as:

$$\text{available} \in \text{SERVICES} \rightarrow \mathbb{N}$$

In the modeling [9] we started by considering that all requested references exist in the set of available services and that this set and the set of requests are given in an up-to-date state. We derived by refinement [7] the situation in which we have to take into account new requests, modification of requests, cancelation of requests and update of services time availability.

3.1 Refinement under the Fairness Assumption

Bellow we refine the previous specification described by [9] addressing the problem of fairness. The previous specification captures the notion of flow by a set. In fact, it is possible that a request remains always pending and is never satisfied, because there are always other requests which are processed.

Our solution is to add a priority to each request that is increased the longer it waits and to satisfy the request with the highest priority. If a request is waiting too long (more than a specific deadline) the request is canceled. The event *new_request* gives each new request a priority using a parameter p ($p \in \mathbb{N}$). The variable *priority*, $\text{priority} \in \text{requests} \rightarrow \mathbb{N}$, records the priority of the request and the new condition strengthens the guard of the event *satisfy_request*:

$$\begin{aligned} \forall p \cdot (p \in \text{requests} \wedge \text{status}(p)=\text{pending} \wedge \\ \text{duration}(p) \leq \text{available}(\text{reference}(p)) \\ \Rightarrow \text{priority}(p) \leq \text{priority}(r)). \end{aligned}$$

In order to manage the waiting time of a request we add a time stamp recorded in the variable *timer*, $\text{timer} \in \text{requests} \rightarrow \mathbb{N}$. The event *new_request* gives each new request a time stamp using the variable *clock*, $\text{clock} \in \mathbb{N}$, that grows larger as each successive operation is invoked:

$$\text{timer}(r) := \text{clock}.$$

The event *cancel_request* is enabled if the difference between the current clock value and the time stamp of the request is larger than the deadline of the request:

$$\text{clock} - \text{timer}(r) > \text{deadline}(r),$$

where the constant *deadline*,

$$\text{deadline} \in \text{REQUESTS} \rightarrow \mathbb{N},$$

records the maximum time for a request to be processed. The event *modify_request* increases the priority of the request if it took longer than the amount of time given by the constant *oldies*,

$$\text{oldies} \in \text{REQUESTS} \rightarrow \mathbb{N}$$

$$\text{clock} - \text{timer}(r) > \text{oldies}(r).$$

The context of the described refinement together with the added variables are given in Fig. 1. Invariants for the variables *timer*, *clock* and *priority* together with the corresponding initialization are given in Fig. 2.

The event *satisfy_request* is given in Fig. 3.

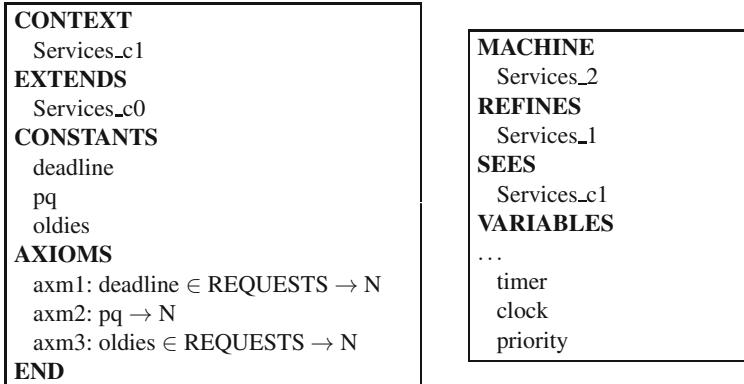


Fig. 1 The context and variables for the refinement

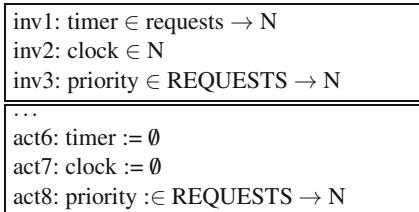


Fig. 2 The invariants and initialization for the refinement

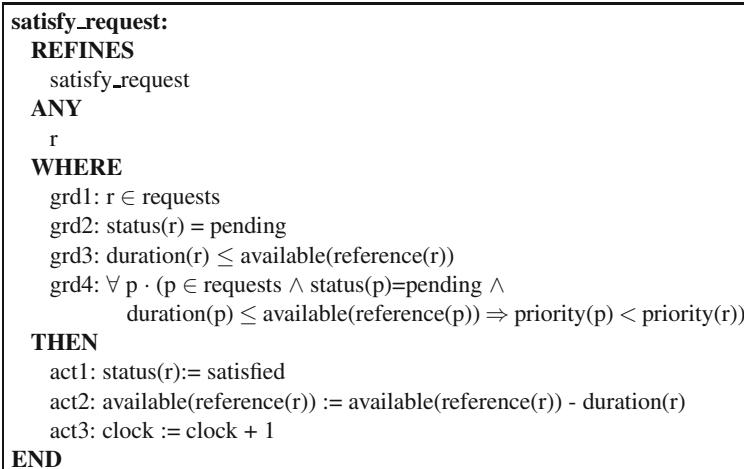


Fig. 3 The *satisfy_request* event

The events *new_request* and *cancel_request* are given in Fig. 4 while *modify_request* is given in Fig. 5. The events *request_available* and *release_available* are not further refined. They have the same specification as in [9].

new_request: REFINES <i>new_request</i> ANY <i>r</i> <i>d</i> <i>s</i> <i>p</i> WHERE <i>grd1: r ∈ REQUESTS \ requests</i> <i>grd2: d ∈ N1</i> <i>grd3: s ∈ SERVICES</i> <i>grd4: p ∈ N</i> THEN <i>act1: requests := requests ∪ {r}</i> <i>act2: status(r) := pending</i> <i>act3: duration(r) := d</i> <i>act4: reference(r) := s</i> <i>act5: priority(r) := p</i> <i>act6: timer(r) := clock</i> <i>act7: clock := clock + 1</i> END	cancel_request: REFINES <i>cancel_request</i> ANY <i>r</i> WHERE <i>grd1: r ∈ requests</i> <i>grd2: status(r) = pending</i> <i>grd3: clock - timer(r) > deadline(r)</i> THEN <i>act1: requests := requests \ {r}</i> <i>act2: status := {r} << status</i> <i>act3: duration := {r} << duration</i> <i>act4: reference := {r} << reference</i> <i>act5: priority := {r} << priority</i> <i>act6: timer := {r} << timer</i> <i>act7: clock := clock + 1</i> END
---	---

Fig. 4 The *new_request* and *cancel_request* events

modify_request: REFINES <i>modify_request</i> ANY <i>r</i> WHERE <i>grd1: r ∈ requests</i> <i>grd2: status(r) = pending</i> <i>grd3: clock - timer(r) > oldies(r)</i> THEN <i>act1: priority(r) := priority(r) + pq</i> <i>act2: clock := clock + 1</i> END
--

Fig. 5 The *modify_request* event

3.2 System Evaluation

The proof statistics given in Table 1 show that 52 proof obligations were generated by the Rodin platform [3], [2]. 47 proof obligations were discharged automatically while the others were discharged by interactive proofs. In Table 1 Services_0 represents the abstract model; Services_1 represents the refinement described in [9] and Services_2 represents the refinement under the fairness assumption.

Table 1 The statement of the development

Element_name	Total	Auto	Manual	Reviewed	Undischarged
Services	52	47	3	0	0
Services_0	25	20	3	0	0
Services_1	4	4	0	0	0
Services_2	23	23	0	0	0

4 Conclusions and Future Work

In this paper, we have presented a specification and verification technique of a multi-agent system for requesting services under the fairness assumption. The informal specification of the system is translated into the Event B notation to verify the required properties. The model refinement that Event-B emphasizes simplifies proofs by providing a progressive and detailed view of the system.

As future work we intend to use UML-B, specified in [4], to model the system and translate the specifications into Event-B for verification. Since the final target is to have an executable code we intend to generate Java code from the Event-B specification with the aid of EB2J plug-in, [1].

Acknowledgements. The work has been funded by Project 264207, ERRIC- Empowering Romanian Research on Intelligent Information Technologies/FP7- REGPOT-2010-1 and the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POS-DRU/89/1.5/S/62557.

References

1. EB2J Tool (2013), <http://eb2all.loria.fr/>
2. Rodin Platform (2013), http://wiki.eventb.org/index.php/Rodin_Platform
3. Rodin User's Handbook v.2.5 (2013), <http://handbook.eventb.org/current/html/index.html>
4. UML-B (2013), <http://wiki.eventb.org/index.php/UML-B>
5. Abrial, J.-R.: The B book. Cambridge University Press (1996)
6. Abrial, J.-R.: Modeling in Event-B: System and Software Engineering. Cambridge Press (2010)

7. Abrial, J.-R., Cansell, D., Méry, D.: Refinement and reachability in event-B. In: Trehanne, H., King, S., Henson, M., Schneider, S. (eds.) ZB 2005. LNCS, vol. 3455, pp. 222–241. Springer, Heidelberg (2005)
8. Sun, J., Liu, Y., Dong, J.S., Wang, H.H.: Specifying and verifying event-based fairness enhanced systems. In: Liu, S., Araki, K. (eds.) ICFEM 2008. LNCS, vol. 5256, pp. 5–24. Springer, Heidelberg (2008)
9. Negreanu, L., Mocanu, I.: Formal verification of service requests in a multi-agent system using Event-B method. In: 8th Workshop on Workshop Knowledge Engineering and Software Engineering (KESE 2012), ECAI 2012, Technical Report TR-2012/1, Montpellier, France, August 27-31, pp. 62–65. University of Almeria, Almeria (2012)

A Survey of Adaptive Game AI: Considerations for Cloud Deployment

Gabriel Iuhasz, Victor Ion Munteanu, and Viorel Negru

Abstract. Modern video games have become an important part of AI research in the past years, largely thanks to the characteristics of their environment and the challenges they pose to AI researchers. This paper is a survey of the current game AI state of the art and highlights important achievements in this field. An adaptive multi-agent system that can be deployed on a cloud infrastructure to solve computational constraints of advanced machine learning methods is also presented.

Keywords: Artificial Intelligence, Video Games, Machine Learning, Multi-Agent Systems, Cloud Computing.

1 Introduction

In recent years, researchers have become ever more active in using games as a test bed for advancing artificial intelligence (AI) techniques, especially due to the fact that games provide a controlled environment while still maintaining properties similar to real-life.

Games are populated with entities called non-player characters (NPCs) with which the player can interact and which can be considered agents. In most games there is a heterogeneous distribution of agent types, some having only reactive agents while others needing proactive ones as part of their AI.

As games usually encompass one or more NPCs, we can infer that games are multi-agent systems (MASs) which are structured hierarchically based on the tasks each agent has to solve. While commercial games have a tendency to stick with simple static methods for their AI systems, academic AI systems do not and mostly

Gabriel Iuhasz · Victor Ion Munteanu · Viorel Negru
West University of Timișoara, Timișoara, Romania
e-mail: {iuhasz.gabriel, vmonteanu, vnegru}@info.uvt.ro

Victor Ion Munteanu · Viorel Negru
Institute e-Austria Timișoara, Timișoara, Romania

utilize adaptive machine learning (ML) techniques. There are two types of ML techniques in game AI: offline and online.

This research is important as lessons learned from developing MAS for game AI can be transferred to real life situations that have similar task distribution and environmental constraints. This paper presents a survey on adaptive agent and multi-agent based game AI solutions. The focus is on strategy games and massive multiplayer online (MMO) as these genres provide a greater challenge and encompass the majority of AI challenges present in other titles. Furthermore, a novel multi-agent solution for deploying ML techniques in games on a cloud infrastructure is presented.

2 Major Trends in Game AI

Generally speaking most modern video games have extremely complex environments for NPCs to interact in [1]. These environments posses incomplete information, each NPC knows only parts of the game state. The level of interaction each NPC has with the environment is also an important characteristic as some modern games each player has unlimited opportunity to interact with the environment, concurrent moves being permitted.

In strategy games there is also a level of asymmetry as one player may have some form of advantage, be it in resources or unit numbers, over the opposing players. Games have more in common to real life scenarios such as military command or disaster management than traditional board games. It should be noted that game environments are also stochastic.

Because of these characteristics, researchers have highlighted the potential to develop "human-level" AI [2]. In his paper Michael Buro [3] identifies some of the major AI challenges present in games, giving good insight into the complexity of the modern game domain. These include resource management, decision making under uncertainty, spatial and temporal reasoning, collaboration, opponent modeling, learning and adversarial real-time planning. Some of these challenges have been actively pursued by the AI community while others have been sadly neglected. The field of game AI collaboration has received little attention.

One important distinction between the academic definition of AI and the gaming industry definition of AI is the fact that game AI has to be entertaining. This means that most NPC behaviour present in games is not based on academic AI techniques but rather on rule of thumb. Sophisticated ML techniques are rarely used. The gaming industry prefers static techniques such as Finite State Machines (FSM), simple Decision Trees, scripting and Rule Based systems.

Hard-coded techniques have one major drawback: they become predictable after a relatively short amount of time and are hard to maintain and debug when they are used to model more complex behaviours. For example, FSMs transitions and the number of states (s) grow exponentially with the number of events (e): $s = 2^e$, consequently increasing the number of transitions (a) even faster: $a = s^2$.

Rule-base Systems are brittle when faced with a problem that is out of bounds with their knowledge base. They are unable to rely on past experience to select a similar rule or update their rule base. Expert knowledge is used in their creation which, once in place, can not be modified and requires substantial effort to maintain and debug. The A* pathfinding algorithm and FSM for decision making make up more than 2/3 of current game AI systems [4].

It is common for AI systems and gameplay elements to be designed and implemented in parallel. This means that if a gameplay mechanic is changed in the latter stages of development it has a catastrophic effect on the AI systems. This situation can be remedied by adaptive and ML techniques which are far more flexible and would require little to no modification. An ML framework could be used in several game titles thus lowering development time and cost.

2.1 Game Genres

Strategy games such as Real-Time Strategy games (RTS) are one of the more challenging genres when it comes to their AI systems. One of the main factors when designing the AI for RTS games is the extremely large state space and environment which are: partially observable, deterministic, sequential, dynamic and continuous. It is also important to note that these types of games posses a massive state space approximated to 10^{11500} in the case of RTS [5]. Also, the number of actions available to the player is also superior in the case of RTS as it has approximately 1 million possible actions while chess has 30. Many sofisticated methodologies an techniques have been used to create AI systems for RTS games. These include: Cognitive architecture [6], Goal-driven autonomy [7], Reactive Planning [8] and Case-based reasoning [9].

FPS (First person shooter) are at the cutting edge when it comes to graphics technologies. It has also benefited from some extremely interesting AI solutions like the AI Director in the game Left 4 Dead¹. Its role is to dynamically adjust gaming difficulty and story pacing based on current player stress levels. Other commercially available titles such as FEAR and Killzone have adopted interesting adaptation mechanisms to their game AI. FEAR uses a STRIPS-planner while Killzone 2 uses HTN (Hierarchical Task Network) planner [10].

ML based systems for FPS are also present in the academic literature as the GDA system which was used Choi [11]. Genetic algorithms and neural networks were shown to manage the complexity of FPS environment [10]. There are still a lot of open research problems: dynamic terrain analysis, fast tactical path finding, efficient combat reasoning, opponent modeling and squad coordination. Some effort has been made in these directions, of particular note being the architecture developed by Hartley [12] which uses incremental dual-state representation and k-d tree-based techniques.

RPG (Role playing game) gameplay involves controlling one or more characters in order to solve quest-based challenges. In contrast to other genres, neutral and

¹ <http://aigamedev.com/premium/article/procedural-director/>

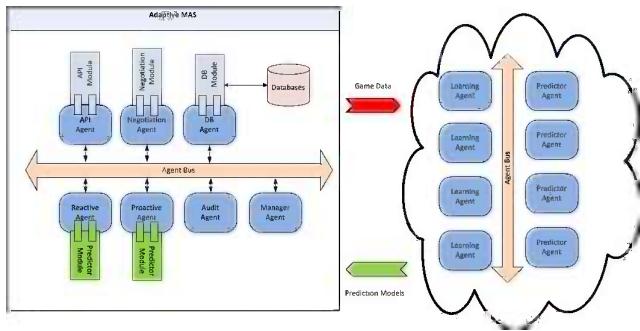


Fig. 1 Proposed Cloud deployment for Game AI MAS

even friendly NPCs are common. Due to the idiosyncrasies of RPG games, content generating techniques have been widely used, research being focused on procedural content generation. In [13] Grey J. and Bryson J. present an AI system capable of dynamically generating dialog with context generated from episodic memory and emotional bias towards past social interactions.

3 Cloud/Server Side AI

Massively multiplayer online (MMO) games feature support for a large number of players which simultaneously interact in a persistent environment. In order to play these games, a player has to log into a server which hosts the environment of the game and its the AI, meaning that on the client side little to no persistent information is stored.

Some MMOs have millions of subscribers and, because of this, the game world is split up into realms or shards, each of them representing a particular instance of a world. This is not the only solution as some MMOs have one cluster of servers that runs the entire game world. Such is the case with EVE online which can host up to 60000 player on a single game world instance.

When it comes to the AI methods used in MMO games these are not sophisticated. Utilizing hard-coded static methods such as those presented in Section 2. The work of Claypool M. [14] shows that the impact of network latency and the degree of lag has varying impact on the gameplay. Games that need only the first level of reasoning (micromanagement or fine unit control) are extremely susceptible to game lag. Other games such as RTS games where micromanagement is only a small part of the AI job are more resilient.

There is a wide variety of MMO sub-genres, the most prevalent one being RPGs (MMORPG). Research done in this area is focused on rule based systems [15] and on Bayesian modeling for human players [16]. Aranda G. proposes another approach that defines a MMOG layer where all the games logics and mechanics are implemented and must be solved at run-time [17].

3.1 *Cloud Game AI Deployment Considerations*

The major shortcoming of ML techniques when applied to game AI is that they require a substantial amount of computational resources which are limited on normal gaming hardware. Some data mining approaches for strategy prediction and army clustering have also been done [18]. They used game replays to extract expert human domain knowledge to create predictive models.

We propose a MAS framework for deploying adaptive game AI on a cloud platform. More and more single player games require perpetual internet connection in order to be played. This is a security measure that is part of digital rights management (DRM) system. This represents an opportunity to use existing DRM technologies to benefit gameplay by deploying adaptive agents on a cloud computing infrastructure.

Our system has 9 agent types: API agents that handle interface with game instances, reactive and proactive agents which execute tasks depending on the game genre, audit and manager agents that handle MAS deployment and stability, negotiation agents that handle all inter-MAS negotiations and database agents that handle all database queries, databases varying from shared knowledge ones to game event logging ones.

The most important part of our proposed framework is the two agent types that are the learner agent and the predictor agent deployed in the cloud. The learner agent receives data in the form of past game states and produces a viable predictive model. Once the learning agent finishes, it exports the model to the predictor agent which then proceeds to validate the predictive model. Once this is done the model can be incorporated into the AI subsystem. It is important to note that most ML methods can be encapsulated into learner and predictor agents. By scaling these agents we can address the high computational cost of such methods while still maintaining an adequate QoS.

4 Conclusions

This paper presents a survey on advances and current research in ML and MAS solutions for computer games. This survey focused mainly on RTS and MMO game genres as the authors feel that these represent the biggest challenges when it comes to game AI.

Open research questions have also been highlighted such as inter-agent negotiation and the lack of a game domain ontology to aid with interoperability. A cloud-based MAS solution is proposed in order to alleviate some of the computational constraints of ML techniques applied to game AI systems.

In a field where agent modeling is more mature, standard ontologies have been developed in order to formalize the relevant domain knowledge. There has been little to no work for creating a viable game domain ontology which would be extremely useful for MAS interoperability and reusability.

Acknowledgements. This work was partially supported by the Romanian national grant PN-II-ID-PCE-2011-3-0260 (AMICAS), FP7-REGPOT-CT-2011-284595 (HOST) and by the strategic grant POSDRU/CPP107/ DMI1.5/S/78421, Project ID 78421 (2010), co-financed by the European Social Fund - Investing in People, within the Sectoral Operational Programme Human Resources Development 2007–2013. The views expressed in this paper do not necessarily reflect those of the corresponding projects' consortium members.

References

1. Lewis, J.M., Trinh, P., Kirsh, D.: A corpus analysis of strategy video game play in starcraft: Brood war. In: Proceedings of the 33rd Annual Conference of the Cognitive Science Society (2011)
2. Laird, J.E., van Lent, M.: Human-level ai's killer application: Interactive computer games. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence. pp. 1171–1178, AAAI Press (2000)
3. Buro, M.: Real-time strategy gaines: a new ai research challenge. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI 2003, pp. 1534–1535. Morgan Kaufmann Publishers Inc., San Francisco (2003)
4. Yildirim, S., Stene, S.B.: A survey on the need and use of ai in game agents. In: Proceedings of the 2008 Spring Simulation Multiconference, SpringSim 2008, pp. 124–131. Society for Computer Simulation International, San Diego (2008)
5. Aha, D.W., Molineaux, M., Ponsen, M.: Learning to win: Case-based plan selection in a real-time strategy game. In: Muñoz-Ávila, H., Ricci, F. (eds.) ICCBR 2005. LNCS (LNAI), vol. 3620, pp. 5–20. Springer, Heidelberg (2005)
6. Langley, P., Choi, D.: A unified cognitive architecture for physical agents. In: Proceedings of the 21st National Conference on Artificial Intelligence, AAAI 2006, vol. 2, pp. 1469–1474. AAAI Press (2006)
7. Muñoz-Avila, H., Jaidee, U., Aha, D.W., Carter, E.: Goal-driven autonomy with case-based reasoning. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 228–241. Springer, Heidelberg (2010)
8. Josyula, D.P.: A unified theory of acting and agency for a universal interfacing agent. PhD thesis, College Park, MD, USA, AAI3202442 (2005)
9. Szczepanski, T., Aamodt, A.: Case-based reasoning for improved micromanagement in real-time strategy games. In: Proceedings of the Workshop on Case-Based Reasoning for Computer Games, 8th International Conference on Case-Based Reasoning, ICCBR 2009, pp. 139–148 (July 2009)
10. McGee, K., Abraham, A.T.: Real-time team-mate ai in games: a definition, survey, & critique. In: Proceedings of the Fifth International Conference on the Foundations of Digital Games, FDG 2010, pp. 124–131. ACM, New York (2010)
11. Choi, D.: Reactive goal management in a cognitive architecture. Cogn. Syst. Res. 12(3–4), 293–308 (2011)
12. Hartley, T., Mehdi, Q.: Online action adaptation in interactive computer games, vol. 7, pp. 28:1–28:31. ACM, New York (2009)
13. Grey, J., Bryson, J.J.: Procedural quests: A focus for agent interaction in role-playing-games. In: Proceedings of the AISB 2011 Symposium: AI & Games (2011)
14. Claypool, M., Claypool, K.: Latency and player actions in online games. Commun. ACM 49(11), 40–45 (2006)

15. Ballinger, C.A., Turner, D.A., Concepcion, A.I.: Artificial intelligence design in a multi-player online role playing game. In: Proceedings of the 2011 Eighth International Conference on Information Technology: New Generations, ITNG 2011, pp. 816–821. IEEE Computer Society, Washington, DC (2011)
16. Synnaeve, G., Bessière, P.: Bayesian modeling of a human mmorpg player. CoRR abs/1011.5480 (2010)
17. Aranda, G., Botti, V., Carrascosa, C.: Mmog based on mas: the mmog layer. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2009, Richland, SC, pp. 1149–1150. International Foundation for Autonomous Agents and Multiagent Systems (2009)
18. Synnaeve, G., Bessière, P.: Special tactics: A bayesian approach to tactical decision-making. In: CIG, pp. 409–416 (2012)

From Agent-Oriented Models to Profile Driven Military Training Scenarios

Inna Shvartsman and Kuldar Taveter

Abstract. We propose an approach for creating agent-based “man-in-the-loop” simulation scenarios for training military and paramilitary staff. The approach is based on psychological theories and enables to define small standalone simulation scenarios for a certain context. It considers several types of personality profiles. Each profile is represented as a combination of needs-based personality characteristics. The overall objective of this research is to achieve realistic “man-in-the-loop” military training scenarios where some roles are played by humans and some other roles by software agents.

Keywords: Agent-oriented modeling, military trainings, agent simulations.

1 Introduction

Personality traits are the unique sets of attributes possessed by individuals. In psychology, trait theory is an approach to the study of human personality. Personality generally refers to the character of an individual or his/her permanent behavioral traits [1]. According to recent studies [2], in a military task environment, a very important role for soldiers’ situational perception is played by two narrow need-based personality traits: Sensation Seeking and Need for Structure. Sensation seeking is a personality trait defined by the tendency to search for experiences and feelings that are “varied, novel, complex and intense” [3], and by the readiness “to take physical, social, legal, and financial risks for the sake of such experiences” [3]. Personal need for structure is another personality trait defined by a desire for certainty and clarity, and a corresponding aversion to ambiguity [4].

Inna Shvartsman · Kuldar Taveter
Tallinn University of Technology, Department of Informatics,
Akadeemia tee 15a, 12618 Tallinn, Estonia
e-mail: innashvartsman@hot.ee, kuldar.taveter@ttu.ee

In this paper we propose an agent-oriented modeling approach for designing and conducting military training scenarios. Our approach is based on psychological theories. It considers several types of personality profiles. Each profile is defined in terms of the individual set of skills, such as reaction speed and completeness of activities, and team skills, such as attention on the activities by other team members and helpfulness towards other team members. We also aim to map each profile to the scale of needs-based personality characteristics with Sensation Seeking on one end and Personal need for structure on another [2]. Our objective is to achieve realistic “man-in-the-loop” military training scenarios where some roles are played by humans and some other roles by software agents.

In our approach, we first represent each training scenario by a set of agent-oriented models described in section 2 and then define for each scenario separately software agents with different psychological profiles, based on the models. The resulting software agents are guided by simple rules that are defined based on the descriptions of the psychological profiles of interest and evaluation criteria for the scenario. The profiles and the corresponding agents differ in the levels of the following criteria: reaction speed, completeness of performing an activity, attention on the activities by other team members, and helpfulness towards other team members. We are interested in the overall emergent behavior of the simulation system consisting of humans and software agents performing the scenario as a whole rather than in mimicking as precisely as possible human behaviors.

2 Agent-Oriented Modeling

In our approach, training scenarios are defined by agent-oriented modeling (AOM). AOM [5] is a top-down approach for modeling and simulating the behaviors of socio-technical systems. In the problem domain addressed by us, a socio-technical system is a “man-in-the-loop” military training system. In AOM, a problem domain is first conceptualized in terms of the goals to be achieved by a socio-technical system, the roles required for achieving them, and the domain entities embodying the required knowledge. The roles are mapped to the agents playing the roles, the goals – to the activities performed by the agents, and the domain entities – to the items of knowledge held by the agents. Models are considered as abstractions. Appropriately abstracting a system can reduce its complexity for better understanding of the system’s particular aspects and their impact on its behavior. The types of models that are relevant for this paper are goal models and role models.

A goal model can be considered as a container of three components: goals, quality goals, and roles [5]. A goal is a representation of a functional requirement of the socio-technical system, that is, a training system. A quality goal, as its name implies, is a non-functional or quality requirement of the system. Goals and quality goals can be further decomposed into smaller related sub-goals and sub-quality goals. Goal models also determine roles that are capacities or positions that are needed to achieve the goals. Role models describe the capacities or positions that are required for achieving the goals.

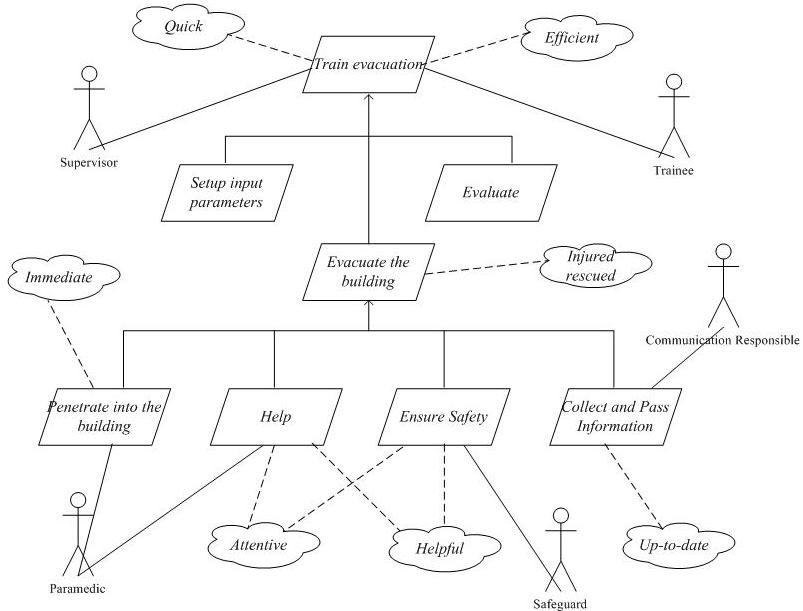


Fig. 1 The high-level motivation model for evacuation training simulation

Fig. 1 represents the training scenario, based on an example from our earlier research [6], by means of a goal model. Goals are represented as parallelograms, quality goals are clouds, and roles are stick figures. The arcs indicate relationships between constructs. The purpose of the scenario is to train evacuation.

In our approach, simulations are tuned by quality goals. Our simulation scenario is characterized by such quality goals as Immediate, Attentive, Helpful and Up-to-date – as is illustrated by the lower part of Fig. 1. Both quality goals characterize the behaviors of agents playing the roles Paramedic and Safeguard.

3 Proactive vs. Reactive Behavior in Training Scenarios

This section describes how the behaviors of software agents can be defined based on different psychological profiles. Software agents are characterized along the dimension of proactivity vs. reactivity. Acting in advance of a future situation, rather than just reacting, is understood as proactive behavior. In terms of agent's context, according to [7], proactivity can be defined as follows: “agents do not simply act in response to their environment; they are able to exhibit goal-directed behavior by taking the initiative”. In case of reactivity, “agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it.” In the military context, where the responsibilities

and constraints of roles are generally well defined by, for example, rules of combat, it is important to build flexible, adaptive leaders with keen understanding and strong decision-making skills. Proactive soldiers and leaders generally do not need to be asked to act, nor do they require detailed instructions.

According to the model of a training scenario represented in Fig. 1, the training exercise is evaluated by the dimensions of speed, efficiency, attentiveness and helpfulness by team members. All of them correspond to the respective quality goals, as can be seen in Fig. 1. Within this paper, we focus on the quality goals of attentiveness and helpfulness by team members because they well correlate with the proactivity of team members [8]. In the training scenario, proactive behavior is particularly required of performers of the roles Paramedic and Safeguard. We therefore focus on these two roles. According to [8], helping behaviors can simply be categorized as reactive helping and proactive helping, whereas reactive helping is triggered by an external request. On the other hand, proactive helping is not initiated by help requests but by the anticipation of others' needs from shared by the agents' knowledge — even if such needs are not directly expressed [9].

According to [10], a quality goal ("soft goal" as the authors term it) is achieved ("satisfied" as the authors describe it) when thresholds of some precise criteria are reached. In psychological training, the achieving of quality goals is characterized by discrete scale from 0 to 3. Based on the criteria, we can define agents playing the roles Paramedic and Safeguard that meet these criteria to a greater or lesser degree. It is done by representing the corresponding agent behaviors by their behavioral rules in Table 1. These rules include a number of undefined in this paper constructs, several of which are concerned with message exchange between agents and situation awareness by agents. The "Request for help" and "Offer help" constructs denote the respective messages sent and received by the agent in focus. The "Assess the situation" construct denotes assessing the situation with respect to performance by other agents. The "Possible problem" construct denotes the situation where *any* other agent has a problem possibly requiring help by other agents.

Table 1 Prototypical behavioral rules

Quality goal / Scale	0	1	2	3
Attentive	No behavioral rule	ON RECEIVE Request help THEN Assess the situation	ON RECEIVE THEN Assess the situation ON SEND THEN Assess the situation	ON Interaction THEN Assess the situation
Helpful	No behavioral rule	ON RECEIVE Request help THEN WAIT N Sec.; SEND Request help	ON RECEIVE Request help THEN SEND Offer help	ON Possible problem THEN BROADCAST Offer help

Finally, the “Interaction” construct refers to any interaction between any two agents or an agent and its environment in the training system.

According to [8], to help other agents often requires the agent to monitor the performance by other agents. In our approach, we define paying attention through assessing the performance by other relevant agents, relying on the common knowledge by the agents involved. In the training scenario of evacuation under discussion here, this means that an agent playing the role of Safeguard assesses the performance of the agent playing the role of Paramedic and the other way round, based on the shared by these agents situational knowledge. In the second row of Table 1, the behavioral rules corresponding to the quality goal “Helpful” are represented in a similar manner. The most helpful agent is the one that offers help to everyone whenever any of the situation assessments performed by this agent indicates a possible problem with any of the other agents. Attentiveness and helpfulness are separate characteristics that may have different levels for the same agent.

We define for software agents of the training system constants *payingAttention* and *beingHelpful* reflecting the level of attentiveness and helpfulness, respectively. Next, we can define the following logic applied as a part of scenario for achieving the goal “Help” by a software agent playing the Paramedic role:

```

Input the beingHelpful constant
Switch to behavior based on beingHelpful value
Case 0
Break // No behavioral rule
End Case
Case 1
If RECEIVE Request help
WAIT N Sec.
SEND Request help
End If
End Case
Case 2
If RECEIVE Request help
SEND Offer help
End If
End Case
Case 3
BROADCAST Offer help
End Case
End Switch

```

4 Conclusions

In this research, an approach for creating agent-based simulation scenarios for training military and paramilitary staff is proposed. The approach is based on psychological theories. Two different needs-based personality characteristics are discussed:

Sensation Seeking and Personal Need for Structure. Agent-oriented modeling is used to define and visualize training scenarios by goal and role models. In the case study, the training exercise is evaluated by the dimensions of speed, efficiency, attentiveness and helpfulness by team members. All of them are characterized by the respective quality goals. The agents are defined based on the criteria proposed for achieving the quality goals. The corresponding agent behaviors are represented by their behavioral rules. According to the previous and current studies on proactive and reactive behaviors in psychological and military contexts, it can be hypothesized that software agents enacting sensation seekers need to be more proactive, while software agents enacting structure preferrers need to be more reactive. In this research, many aspects were not considered, and a number of research problems were left for future research. Among them is a plan to design a system where software agents following different psychological profiles are generated from agent-oriented models. In our future work the hypothesis stated above needs to be tested in real experiments.

Acknowledgements. This research was supported by the Estonian Doctoral School in Information and Communication Technology.

References

1. Kassin, S.: *Psychology*. Prentice-Hall, USA (2003)
2. Parmak, M., Euwema, M.C., Mylle, J.: Changes in Sensation Seeking and Need for Structure Before and After a Combat Deployment. *Military Psychology* 24 (2012)
3. Zuckerman, M.: Sensation seeking. In: London, H., Exner, J. (eds.) *Dimensions of Personality*. Wiley, New York (1978)
4. Neuberger, S.L., Newsom, J.T.: Personal need for structure: individual differences in the desire for simple structure. *Journal of Personality and Social Psychology* 65, 113–131 (1993)
5. Sterling, L., Taveter, K.: *The Art of Agent-Oriented Modeling*. MIT Press, Cambridge (2009)
6. Shvartsman, I., Taveter, K., Parmak, M., Meriste, M.: Agent-oriented modelling for simulation of complex environments. In: IMCSIT 2010, The International Multiconference on Computer Science and Information Technology, pp. 209–216. IEEE Computer Society (2010)
7. Wooldridge, M.J., Jennings, N.R.: Agent Theories, Architectures, and Languages: a Survey. In: Wooldridge, M.J., Jennings, N.R. (eds.) ECAI 1994 and ATAL 1994. LNCS, vol. 890, pp. 1–22. Springer, Heidelberg (1995)
8. Fan, X., Yen, J.: Modeling and simulating human teamwork behaviors using intelligent agents. *Journal of Physics of Life Reviews* 1, 173–201 (2004)
9. Marks, M., Zaccaro, S., Mathieu, J.: Performance implications of leader briefings and team interaction training for team adaptation to novel environments. *Journal of Applied Psychology* 85, 971–986 (2000)
10. Jureta, I.J., Faulkner, S., Schobbens, P.-Y.: Achieving, Satisficing, and Excelling. In: Hainaut, J.-L., et al. (eds.) ER Workshops 2007. LNCS, vol. 4802, pp. 286–295. Springer, Heidelberg (2007)

An Agent-Based Solution for the Problem of Designing Complex Ambient Intelligence Systems

Marius-Tudor Benea

Abstract. This paper presents an agent-based solution for the problem of designing complex ambient intelligence systems. The solution offered is a method of combining many subsystems, developed individually, into a bigger, more complex, system, meant to make use of the collective intelligence emerging from collaboration of all the subsystems. An example of such a complex ambient intelligence system is the environment built around a smart city, where it is desired to improve the experience of the inhabitants of a city, while still carefully using the available resources.

1 Introduction

Ambient Intelligence (AmI) “provides a vision of the Information Society where the emphasis is on greater user-friendliness, more efficient services support, user-empowerment, and support for human interactions”, as described in [4]. As we can see, the focus in AmI environments is put on the user. But there are some important problems, too, that are caused mainly by the user itself (e.g. the transportation problem and its consequences, caused by the user’s preference for the comfort of a drive from door to door over other means of transportation, which creates problems like traffic congestion and pollution). However, even in such cases AmI can offer an answer. In the case of a smart city, for example, we could use AmI not only as a way to offer users a better experience but also to improve the efficiency and environmental-friendliness of the cities, given the communication channel between the city and the user and vice-versa offered by the ambient intelligence environment. The main problem, in this case, is how to create intelligent enough cities in order

Marius-Tudor Benea
Laboratoire d’Informatique de Paris 6,
University Pierre and Marie Curie, France and
Computer Science Department,
University Politehnica of Bucharest, Romania
e-mail: marius-tudor.benea@lip6.fr

attract the user to use some publicly available and optimized services (e.g. the public transportation system for transiting the city). Further, the information generated by the users could be used to optimize even more these services and to create even more intelligent cities.

While the advantages of having ambient intelligence systems of a city level complexity which posses an increased degree of intelligence are clear by now, it's not very obvious how to develop them. In this paper we present a solution for designing multi-agent systems (MAS) for building ambient intelligence applications characterized by an increased level of complexity, like in the case of smart cities, meant to push the intelligence of such systems to a better level. The approach to AmI that we use is agent-oriented. The solution offered is to combine many small AmI systems, designed, developed and optimized individually in order to solve specific problems, into bigger, more intelligent AmI systems, that are supposed to have high degrees of intelligence in the form of the collective intelligence emerging from the subsystems composing them. Together with this solution, the paper also presents a method for doing it.

The structure of the paper is the following one: Section 2 presents some related works. In Section 3 and its subsections we describe the solution proposed and we discuss the details that concern it. Finally, we draw some conclusions and we present the perspectives of our work, in Section 4.

2 Related Work

Many research projects have been conducted in order to propose solutions for improving the performance and efficiency of complex environments. One particular case of a such environment, that we also use as an example throughout this paper, is the case of the cities. There is a trend of making them more efficient and more friendly to their inhabitants, transforming them, thus, into smart cities.

S-CLAIM (Smart Computational Language for Autonomous, Intelligent and Mobile Agents), [1, 9], offers a good solution for developing AmI applications, as proven in [1]. It forces a hierarchical representation of agents which can be successfully used in order to accomplish important ambient intelligence tasks like context-awareness, information about the context being inferred directly from the architecture of the multi-agent system underlying an AmI system. Based on the CLAIM language [7, 16, 15], which was successfully used in projects like Ao Dai [6], the framework comprising the language and its platform offers expressiveness, ease of use, support for multiple platforms and some other advantages. This work is based on the representation method for AmI applications based on mobile agents, offered by S-CLAIM and it extends it accordingly, considering the needs of the studied problem.

Streitz et al., [14], studies how information and communication technology, particularly ambient intelligence, will influence the future of our cities, considering the tendency to shift towards an Urban Age. These cities of the future are called *humane cities*, defined as “places and environments where people enjoy everyday life

and work, have multiple opportunities to exploit their human potential and lead a creative life”, and are the future form smart cities will take. The accent is put on the fact that such cities will be created by means of ambient intelligence environments.

Many researches related to smart cities have been conducted with the goal in mind of improving the transportation system, which is an important problem in urban areas (but they are limited to it). Passos et al., in [11], recall many of the different aspects involved in the AmI concept and identify potential applications of its technological asset in ITS-related scenarios. Candamo et al., [2], made a survey of the image-processing human behavior recognition algorithms in the context of transit applications. This is of interest for both AmI and ITS, but, however, focuses only on one particular problem that could be further integrated by these systems, namely, the problem of understanding humans in the context of transportation, by means of image processing techniques.

None of the works presented above propose guidelines for designing very complex ambient intelligence systems, like in the case of smart cities, composed of many subsystems, strongly interconnected, that collaborate in order to obtain a better performance (both from a user point of view and from the point of view of the global system - in our case, the city), by means of the intelligence that emerges from their global collaboration.

3 Designing Complex AmI Systems Using MAS

In this section we present the approach for designing complex AmI systems, like the example of a city (smart city), which is used as a case study. In few words, the global system is considered to have a hierarchical structure, being a collection of subsystems, each of them designed to solve one particular problem. These subsystems are further composed of other components (in the form of subsystems), until they reach an atomic system, individually built in order to implement a simple scenario. The global AmI system is a system of systems and one of its main characteristics is the collective intelligence emerging from the collaboration between all the subsystems composing it, which is higher than the sum of the degrees of intelligence of all components.

Given the distributed character of the AmI systems and the continuous changes of their execution context and topologies, an approach to AmI based on multi-agent systems is an excellent solution, [1, 10, 5, 6, 13, 12]. The current approach is, thus, agent-based and the solution used is inspired by the agent representation approach used by the S-CLAIM language, [1, 9], in which the agents are grouped into simple hierarchies (where simple means that each agent has at most one parent) and it extends this approach for more complex scenarios.

Considerations. The current approach is exemplified, as mentioned above, on the case study of a smart city, but it could be applied to any other scenario where we deal with complex environments. In such an environment, an AmI system meant to help the user in his daily activities should take into consideration a lot of aspects.

For a small AmI application, like in the case of the *SmartRoom* scenario, [1], it is easy to have an overview of how the system will look like and it's quite obvious how to design it. It is difficult, however, to design, a-priori, an AmI system that takes into account all the possible problems and scenarios occurring in a city, using the same method as for the case of the more simple *SmartRoom* scenario. When we discuss about cities (or even bigger environments), no single developer or team could approach all the possible problems in an optimal way. Thus, a better solution is to consider that stand-alone AmI applications have been developed and optimized for simple scenarios and that a good method of integrating them is available. Considering the existence of a framework like S-CLAIM and its associated platform, [1], which offers a good development environment for the implementation of a wide range of AmI applications and by using which all these applications would have similar structures and execution specifications, their integration is possible.

Given these considerations, the problem of designing complex AmI environments reduces to the problem of defining a method for combining multiple stand-alone AmI applications, giving birth to bigger environments that make use of the ingenuity of each subsystem and its components in a clever way. As the multi-agent systems on which the AmI applications developed with S-CLAIM on top of its associated platform, [1], are based on well defined topologies (in the form of the agent hierarchies described in the same article), the problem becomes the one of extending S-CLAIM's formalism with a method for growing the underlying agent graphs of multiple applications into bigger graphs, by means of some well defined techniques adapted to the context of complex AmI systems.

Analysis of a Smart City Example. One of the first problems that have to be solved when designing a city level AmI system is how to divide it into components that could be managed by separate AmI applications. There is no best answer for this. One option would be to consider a hierarchical physical structure, starting from the user and the devices he uses and continuing with his places of interest (home, work, etc.). These places of interest can be grouped together into bigger entities, like neighborhoods, that group into districts which, eventually, compose the whole city. Taking the city of Paris as an example (this choice is simply a preference of the authors; any other city or even a fictitious one could be considered), we can see how it is divided into districts in Figure 1, left.

We have now a division of Paris from a physical point of view, by districts. So, we can consider that each such division is represented by an AmI environment, which works well for the respective region. However, we can see that in the example from the previous paragraph most of the elements considered are static. But there is one, the user, who is also characterized by mobility. And he is not the only one. Let's consider the cars. They are mobile too, so they can change their position. They can travel from one district to another. In Figure 1, right, we can see how, adding the transport network (only the métro network is considered) over the map from the left side of the figure, the districts aren't isolated anymore. They become interconnected.

In order to have a simple hierarchy of agents, as in [1], in which each agent has only one parent, we can make use of agent mobility in order to deal with the problem

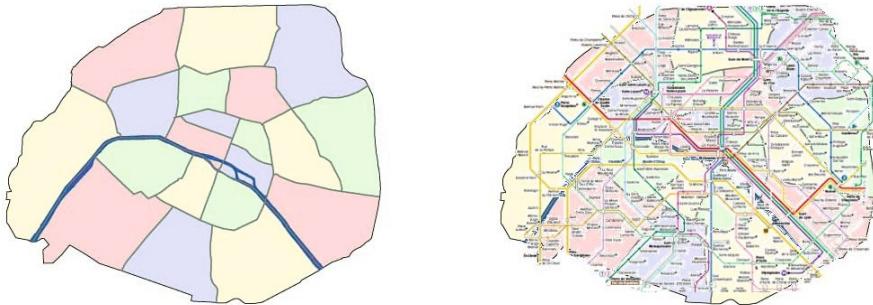


Fig. 1 Division of Paris into districts (left & right) + métro network (right)

of the mobile components of the environment. So, for example, let's consider a bus, the bus number 38. It has to pass from the 1st district to the 6th (it's not important where exactly they are on the given map). If it is considered to be located in one of the districts of Paris and if we want to express this as part of the context described by the agents' hierarchy, we can simply move the agent that assists the bus from the sub-hierarchy of one district to the sub-hierarchy of the other. Thus, mobility is a simple solution for this problem.

What if we consider a particular 38 bus as the child of the 38 bus line (in each situation, by mentioning a certain entity we refer to the agent representing it), and the 38 bus line as the child of the Paris bus network, also a child of agent managing the Paris transportation system, etc.? In this situation there is no direct way to represent the district where the bus is located. So, a pure hierarchical structure (i.e. each agent has only one parent) of the multi-agent system underlying the AmI system of the city is not a solution. There is also a need for communication between the AmI subsystems of a certain level. For example, an agent located physically in the 6th district should know that there is a 38 bus in another district that is 15 minutes away from the entity it represents and that it is the closest 38 bus.

The problem of the bus can be solved simply by adding, for example, one agent to each bus station (it is also a good thing to consider for a transportation AmI system, as the stations possesses useful information about the conveyances that pass through them). The station knows when a bus passed so it can inform the line's agent about this, so that the line has an overview of the positions of all its buses. However, this can be expressed in a better way too, also encoding it in the hierarchy of agents, but this time a more complex hierarchy, in which an agent can have more than one parent. For example, a station could have as parent the district it is located in and also all the bus lines that contain that station. The possibility of having more parents is not supported in S-CLAIM, but it is undoubtedly an useful feature.

What properties would such a complex hierarchy have? The most important one is that it is possible to decompose it into several simple ones, in which the agents having more parents would be copied in each simple hierarchy, according to one single child-parent link. So, if we can decompose a complex system this way, can we also reverse the process and use the resulted method to combine different AmI

```

1   repeat
2       choose two AmI applications and the multi-
3           agent systems  $A$  and  $B$  behind them;
4       find the set  $T$  of all the pairs of twin agents
5           in the two multi-agent systems,  $A$  and  $B$ ;
6       for each pair  $(x_a, x_b) \in T$  of twin agents:
7           create a new agent,  $x_{ab}$ ;
8            $parents_{x_{ab}} := parents_{x_a} \cup parents_{x_b}$ ;
9            $parents_{x_a} := \{x_{ab}\}$  and  $parents_{x_b} := \{x_{ab}\}$ ;
10           $x_{ab}.open(x_a)$  and  $x_{ab}.open(x_b)$ .
11      define and create a new agent,  $ab$ ;
12      change the parents lists of the root agents of
13          the two AmI systems chosen at step 2 from
14          null to  $\{ab\}$ ;
15      solve further integration problems (e.g. adapt
16          the GUIs).
17 until only one AmI system is left.

```

Fig. 2 Algorithm for building complex, intelligent, AmI systems in a bottom-up manner

systems into bigger, more complex, subsystems? The answer is yes. We can propose a method to identify the agents that can be combined into a single one, having all the capabilities of the comprised agents and the list of all their parents. We can extend this idea in order to provide an alternative to the top-down, rigid, design of an AmI environment. We can do this in a simple, bottom-up, manner, in which smaller systems are separately created according to some immediate and important needs by developers who have a vision over them, and then they are combined in order to form bigger systems.

One other advantage of the model in which each agents can have more parents, beside the possibility to build complex AmI systems in a bottom-up manner is the possibility to find more complex solutions to the problems the agents deal with, by making use of the collective intelligence of all the components that were previously combined in the development process. In other words, an agent can ask all its parents a question and, afterwards, it can aggregate all the answers and draw a conclusion based on them all. This is done directly, quickly and in a parallel manner, without stressing the agent that is in the top of a simple hierarchy, risking thus to create a bottleneck.

The Solution. In the context of complex systems, like the smart cities, we have many, smaller, subsystems. Each subsystem tries to solve one different problem, like in the case of a division of the city according to the services offered to the inhabitants (transportation, events, weather state and forecast, etc.). In order to make these subsystems communicate, a component (let's say a user) should be represented in more such subsystems by the same entity (e.g. the assistant of a user can be a reporting entity for the weather service and part of a group traveling with the same mean of transportation, in the same time). So, the best way to connect the separate

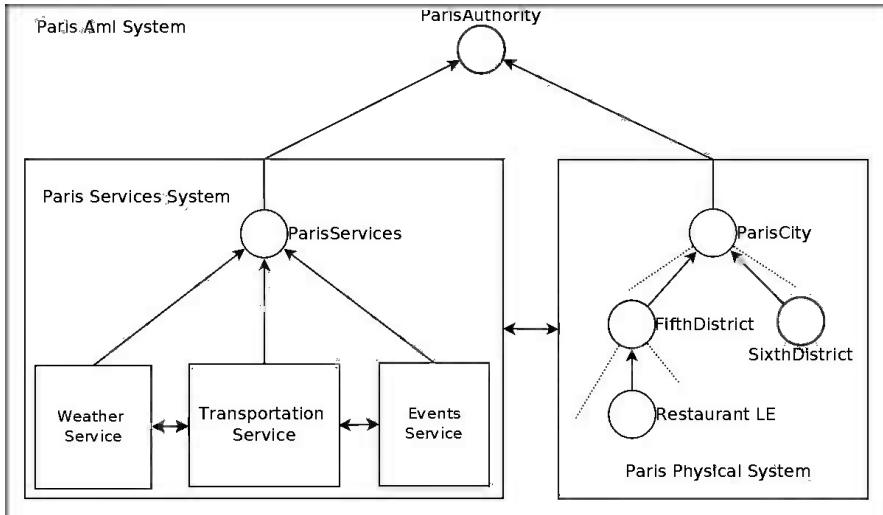


Fig. 3 Example AmI System of Paris. The horizontal arrows suggest that the components are also interconnected directly. A rectangle is a system, a circle an agent.

subsystems is to identify the common parts and to represent them by the same entity (an agent that collects all the capabilities of the agents in each such subsystem and all the parents is a good choice). The question is how to do it. For cases like agents that are single in their computational context, [9], the answer is simple. For more complex situations, appropriate algorithms should be proposed.

The method described in this subsection has a recursive character, so that it can be used to connect any systems into higher level ones, like smart cities together creating smart regions, that connected together can create smart groups of regions and so on, until, finally, we can have an AmI system that works at a global level.

In Figure 2 we present the algorithm to be considered in order to build, in a bottom-up manner, the complex AmI systems described before. The steps 2, 3, the *defining* part of 9 and 11 are left to the appreciation of the developers and are to be studied in depth in the future. For the third step, we define as *twins* two agents which, according to some considerations (e.g. computational context, entity represented, design, name, responsibilities, etc.), are good choices for the future joints between the two old AmI systems. parents_x is the set of parents of the agent x ; $x.\text{open}(y)$ is the action by which agent x absorbs agent y . For more details about the *open* primitive, see [1, 15, 3]. At the steps 9 and 10 an agent supposed to manage the new AmI system is created and given a role.

Exemplifying the Solution. Let's consider an AmI system of Paris comprising the components shown in Figure 3 (Obs. 1: the horizontal arrows are the connections made at the steps 4-8 in the algorithm in Figure 2 and the agents above each system are the ones created at the steps 9 and 10; Obs. 2: This system was already built using the algorithm in Figure 2 from the subsystems represented by the inner

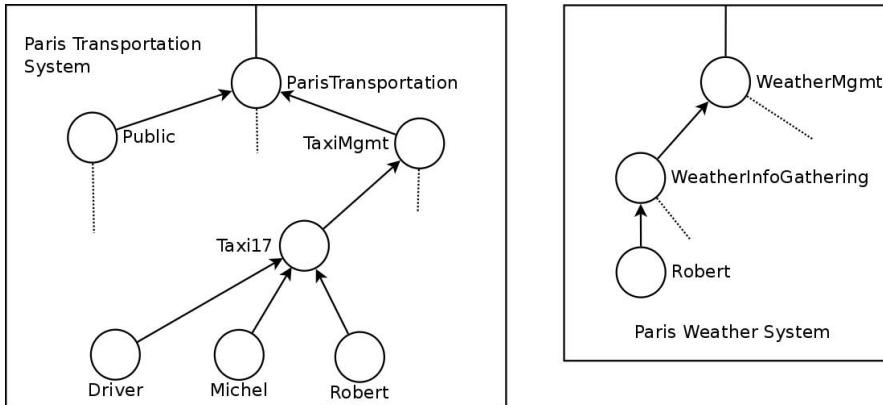


Fig. 4 The transportation and weather services considered separately

rectangles). At the beginning, two users, Michel and Robert, are in the LE Restaurant located in the 5th district of Paris, so their agents are children of the agent representing the restaurant (only the agent managing the restaurant is represented in Figure 3). A taxi comes for the two and both assistant agents become children of the agent managing the taxi, with respect to the transportation AmI application. Robert is still connected to the Internet, so useful information (air pressure, GPS position, etc.) is collected by his mobile device and sent to a separately weather system (Figure 4). We want to connect these two systems into one single system. Thus, the two Robert agents will “compress” into a single agent (Figure 5). Further, the *TransportationAndWeatherMgmt* agent is defined and created in order to manage the new, more complex, AmI system.

Discussion about the Solution on the Basis of the Example. The bottom-up approach for building complex AmI systems, presented in this paper, has a great advantage. It allows the developers to improve and optimize simple, stand-alone AmI systems, in order to offer the best possible solutions for the problems they solve. Thus, the bigger system, which makes use of the intelligence of these stand-alone systems (its subsystems), developed using the method presented in this work, has access to a very powerful resource, a diversified collection of other systems (approaches to problems), which, according to [8], could create a more intelligent whole, as the diversity is a factor that considerably improves the collective intelligence of groups/collections of entities.

As an example, in figures 4 and 5 we have some simplified structures of the systems represented. They include many more agents and each agent may also be part of other hierarchies, as, for example, an agent representing a bus may be the child of the line it represents, but it may also be part of the traffic in the area it is located in (determined using its GPS coordinates). Considering that it wants to find

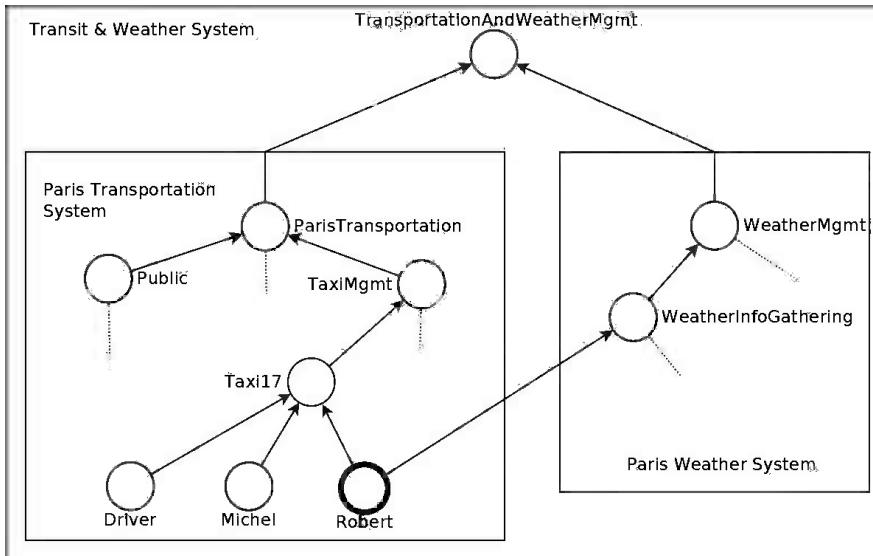


Fig. 5 The transportation and weather services, connected

out the remaining time until a certain destination point (e.g. in order to answer a question put by the assistant of a user who wishes to go to the location of a certain event, from his agenda), it could ask its both parents and each of them will provide an answer, based on their proper techniques. The bus line can use some statistic information about the times needed for the same distance in similar conditions in the past and about the way buses in front of it went earlier. The traffic agent for that area offers some more fresh information about the state of the traffic. However, as the buses use their own lane, the answer provided by the traffic agent may differ from the actual time needed by the bus in order to reach the specified destination. So, it is up to the bus to combine the two pieces of information in order to generate an answer. So the bus could draw a better conclusion as a result of combining the two mentioned systems rather than being part of them, separately. And this is a very important aspect for such a complex and open system like a smart city, as it is possible to have only a partial view of the environment at a given time moment, thus having to deal with uncertainty.

4 Conclusions and Perspectives

In this paper we provided a bottom-up solution for designing complex agent-based ambient intelligence systems, like for the case of smart cities, in which the final system is the result of combining many individually developed agent-based applications designed to deal with smaller problems. We used smart cities as an example

of such complex AmI environment. The solution we proposed is meant to be used while designing complex AmI systems which the user desires to be endowed with a high degree of intelligence.

Some open research problems are how to automatically do the interconnection of the subsystems into a bigger systems (dealing with the 2, 3, 9 and 11 steps in Figure 2), what protocols to use for the communication between an agent and all its parents to be as effective as possible, how to deal with the complex knowledge records obtained by combining multiple answers to a single question and how to measure and use the intelligence that emerges from the collaborations of all the subsystems in order to offer the best possible services to the users.

References

1. Baljak, V., Benea, M.T., Seghrouchni, A.E.F., Herpson, C., Honiden, S., Nguyen, T.T.N., Olaru, A., Shimizu, R., Tei, K., Toriumi, S.: S-claim: An agent-based programming language for ami, a smart-room case study. *Procedia Computer Science* 10(0), 30–37 (2012)
2. Candamo, J., Shreve, M., Goldgof, D., Sapper, D., Kasturi, R.: Understanding transit scenes: A survey on human behavior-recognition algorithms. *IEEE Transactions on Intelligent Transportation Systems* 11(1), 206–224 (2010)
3. Cardelli, L., Gordon, A.D.: Mobile ambients. In: *Proceedings of POPL 1998*. ACM Press (1998)
4. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J.: Scenarios for ambient intelligence in 2010. Office for official publications of the European Communities (2001)
5. El Fallah Seghrouchni, A., Florea, A.M., Olaru, A.: Multi-agent systems: A paradigm to design ambient intelligent applications. In: Essaaidi, M., Malgeri, M., Badica, C. (eds.) *Intelligent Distributed Computing IV. SCI*, vol. 315, pp. 3–9. Springer, Heidelberg (2010)
6. El Fallah Seghrouchni, A., Olaru, A., Nguyen, N.T.T., Salomone, D.: Ao dai: Agent oriented design for ambient intelligence. In: Desai, N., Liu, A., Winikoff, M. (eds.) *PRIMA 2010. LNCS*, vol. 7057, pp. 259–269. Springer, Heidelberg (2012)
7. El Fallah-Seghrouchni, A., Suna, A.: An unified framework for programming autonomous, intelligent and mobile agents. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) *CEEMAS 2003. LNCS (LNAI)*, vol. 2691, pp. 353–362. Springer, Heidelberg (2003)
8. Howe, J.: *Crowdsourcing: How the Power of the Crowd Is Driving the Future of Business*. Business books, Random House (2009)
9. Olaru, A.: A context-aware multi-agent system for ami environments. Ph.D. thesis, University Politehnica of Bucharest & University Pierre and Marie Curie, Paris (2011)
10. Olaru, A., Florea, A.M., El Fallah Seghrouchni, A.: A context-aware multi-agent system as a middleware for ambient intelligence. *Mobile Networks and Applications* (2012)
11. Passos, L., Rossetti, R., Oliveira, E.: Ambient-centred intelligent traffic control and management. In: *2010 13th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 224–229 (2010)
12. Ramos, C., Augusto, J.C., Shapiro, D.: Ambient intelligence - the next step for artificial intelligence. *IEEE Intelligent Systems* 23(2), 15–18 (2008)
13. Sadri, F.: Ambient intelligence: A survey. *ACM Comput. Surv.* 43(4), 36:1–36:66 (2011)

14. Streitz, N.: Smart cities, ambient intelligence and universal access. In: Stephanidis, C. (ed.) Universal Access in HCI, Part III, HCII 2011. LNCS, vol. 6767, pp. 425–432. Springer, Heidelberg (2011)
15. Suna, A.: Un environnement pour la programmation d'agents intelligents et mobiles. Ph.D. thesis, Université Paris 6 - Pierre et Marie Curie, Paris, France (2005)
16. Suna, A., El Fallah Seghrouchni, A.: Programming mobile intelligent agents: An operational semantics. *Web Intelligence and Agent Systems* 5(1), 47–67 (2004)

Agent-Based System for Affective Intelligent Environment

Mihaela-Alexandra Puică, Irina Mocanu, and Adina-Magda Florea

Abstract. Within the Ambient Intelligence research field there is little attention payed to the affective side of the users. However, we believe that it should not be ignored, because emotions have an important role in all human cognitive processes and in their behavior. A smart environment should be able to detect the emotional state of the persons living there and to adjust its answers according to the user affective needs.

With this objective in mind, we address the idea of building a smart museum. Consequently, we have built an agent-based system that aims to respond to the affective component of the visitors of this museum, by guiding them to the gallery that would induce them a pleasant affective state. In this paper we introduce the agent-based system built, but we focus on the first step that must be done in the system life cycle: detect visitor affective state. We do this by reading their facial expressions, using the Kinect Face Tracking SDK.

1 Introduction

Affective Computing and Ambient Intelligence are both recent fields of study in the Artificial Intelligence research domain. An important amonunt of work was done in both directions, but less interest was shown towards combining them.

We believe the two research fields can help eachother. Ambient Intelligence aims at building smart environments where devices help people in their activities in a non-intrusive manner. Affective Computing aims at improving human-computer interaction by understanding and responding to user emotions and by displaying believable behavior. The goal of our work is to build an environment where agents anticipate both the practical and the affective user needs.

Mihaela-Alexandra Puică · Irina Mocanu · Adina-Magda Florea

University Politehnica of Bucharest, Splaiul Independentei 313, 060042, Bucharest, Romania
e-mail: mihaela.puica@gmail.com,

{irina.mocanu, adina.florea}@cs.pub.ro

In what follows, Section 2 presents related work done in Ambient Intelligence and Affective Computing, focusing on emotion recognition from facial expressions. Section 3 shows the system and the scenario in which the system is applied, Section 4 presents the method and the results obtained and Section 5 draws conclusions and future work.

2 Related Work

Research in Ambient Intelligence emphasizes the user-centric perspective, where various devices capture context information, which is then used by the system to react to changes or to proactively give information to the users by anticipating their needs. Examples include SmartDrawer [2], a system that takes care that the patient takes the pills as advised; Camile [9], dedicated to elderly or disabled people, giving them the possibility to control the lightning in a room; AoDai [8], a multi-agent system that uses context-awareness to guide a user in a smart environment.

The systems mentioned above do not take into consideration user affective state, but there is some work done in this direction. The review in [4] includes systems that infer user affective state and use it to keep the user in a productive state. BioStories [14] builds a model of the user affect which is then used by the system to adapt to the user emotional state.

The primary activity of these systems is to automatically recognize human emotions. There are several methods to do this, resumed by Rafael Calvo and Sidney D'Mello in their review [3]. These methods depend on the forms of the sentic modulation, i.e. the influence of emotion on bodily expression [13].

Facial expressions appear when the face muscles move, resulting in different forms of the eyes, brows, nose and lips, as well as specific wrinkles. In psychology, Paul Ekman did a lot of research, developing a system for objectively classifying expressions (Facial Action Coding System [7]) and arguing that there are 6 basic emotions, universally expressed and recognized by humans in different cultures. In artificial intelligence, research focuses on classification of a small number of emotions using neural networks, genetic algorithms or support vector machines. According to [10], current emotion recognition systems using facial expressions recognize between 2 to 7 emotion categories, with an accuracy of 83% - 98%.

Gestures and posture refer to the whole body and arms position. Both sitting and standing bodies are revealing for user emotions. The Labanotation [1] uses the articulations of the body to determine the position of a standing user. For sitting body postures, The Body Pressure Measurement System (BPMS) [12] recognizes the interest level with an accuracy of about 82%.

Speech is characterized by parameters like pitch or volume, which can be used to determine the emotion encoded into the voice intonation. Current systems for affect detection from voice use 10 to 20 features and have an accuracy of about 80% (see [5] or [15]).

Physiological parameters require special devices to be measured, but the results are encouraging. [3] compared 12 studies that classify 3 up to 8 emotions with an average accuracy of 80%.

Multimodal emotion recognition uses several of the methods presented above to detect affect. It can be done at three levels (data fusion, feature fusion, decision fusion), depending on the moment of result integration. There are few implemented systems and they do not have very conclusive results.

3 An Agent-Based System for Dealing with Affective Behavior

Considering an arts museum, we would like to design an agent that can anticipate the type of art objects that a visitor is likely to appreciate. Based on the user preferences, the agent should be capable to guide them to the gallery that would provide them with a pleasant experience.

In order to do this, the visitor is to be shown samples from each gallery and the agent must read their facial expression, inferring whether the artworks produce a positive or negative effect upon them. Using this information, the agent will recommend the gallery that it considers the most appropriate for the tracked visitor.

Fig. 1 shows the system structure. In the center lies the user because the whole system works for their benefit. Thus, when the visitors enter the museum, it is likely that while looking at the sample objects in the entrance hall, their faces will express emotions like curiosity or dislike. We are only interested in these emotions and we assume that they are strong enough to surpass the affective state of the visitor before entering the museum.

Based on the emotions felt by the visitor, and using the information in the knowledge base and in the feedback dataset, the reasoning engine chooses a certain gallery that will be recommended to the visitor.

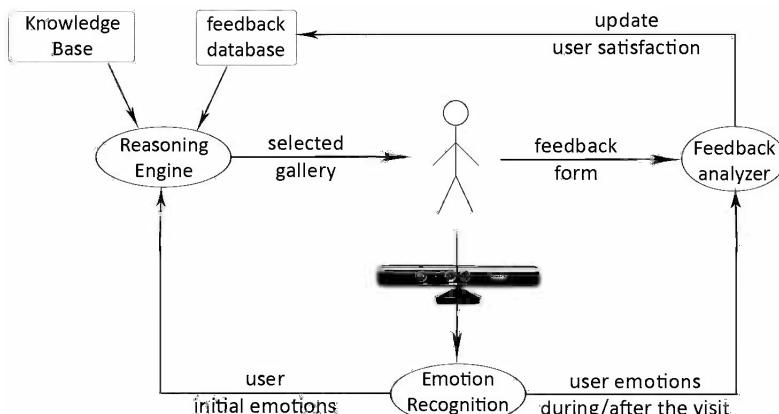


Fig. 1 Agent-based system for smart museum

The knowledge base contains the artworks presented in the museum, each piece being tagged with several keywords that define the type (e.g. painting, sculpture), style (e.g. classicist, renaissance, modernist), theme (e.g. portrait, landscape, battle) and induced state (e.g. melancholy, playfulness, quietness) of that piece of art. Each person has some preferences among these criteria, and the reasoning engine has to infer them based on their initial emotions.

The feedback dataset is initially empty. During the visit, user emotions are again assessed, and at the end of the tour, the visitor is invited to submit their feedback by answering a few, short questions. Both methods are used to assess user satisfaction. This feedback is used to recommend the visitor the next gallery to visit.

In this paper we focus on the first part of the described system, thus the next section details the method we applied to detect visitor emotions.

4 A Method for Emotion Recognition

Our approach was directed towards building a psychological model of human emotion recognition through facial expressions and applying this model on automatic recognition. For that, we first studied the differences in the facial expressions of the six basic emotions, as described by Paul Ekman [6]. After the analysis, we chose to consider only the modifications in the eyes, brows and lips, the nose generally being less revealing for affect detection. Fig. 2 shows the psychological model that we came up with. We can note that happiness and disgust are only expressed through the mouth.

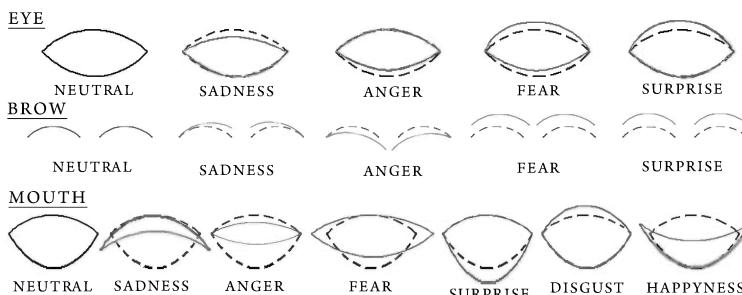


Fig. 2 Face feature modifications for each emotion

For the implementation, we decided to use the Microsoft Face Tracking SDK¹ because it allows us to build applications that can track human faces in real time, which is what we need for a smart museum. This SDK tracks head position and facial features, returning 121 reference points on the face.

Considering the psychological model shown in Fig. 2, we decided to use only 58 points, corresponding to the brows, eyes and mouth. Thus, we have:

¹ <http://msdn.microsoft.com/en-us/library/jj130970.aspx>

- 16 points for the eyes (8 for the left eye and 8 for the right eye)
- 20 points for the brows (10 for the left brow and 10 for the right brow)
- 20 points for the lips (12 for the exterior lips, 8 for the interior lips)
- 2 points for eye centers (left eye center and right eye center)

Fig. 3 shows a representation of the tracked points. At the left of the figure there are the points returned by the Face Tracking SDK, while at the right are those used by our system. The points at the right of the figure were plotted in Octave using feature coordinates obtained through tracking a real person.

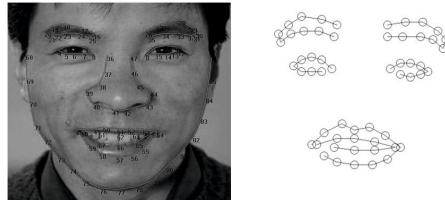


Fig. 3 The points returned by the Face Tracking SDK¹ and the 58 points used by our system

All the experiments using the Kinect sensor were conducted in the Ambient Intelligence laboratory presented in [11].

We followed a supervised learning approach, training a neural network with facial expressions for known emotions. Fig. 4 shows in detail the steps necessary to achieve our goal, from the face tracking up to the emotion output.

After obtaining the 58 points that define the brows, eyes and mouth, we measured 18 distances between the face elements. These are shown in Fig. 5.

At this point, we analysed the data gathered so far to observe how different the values are. Table 1 shows the most frequent values for the distances between the brows and the eyes, computed for one person tracked 6 times, once for each emotion, each dataset containing more than 100 frames.

We can notice that the values for the computed distances are quite similar. To differentiate them better, we propose the following 3 clusters:

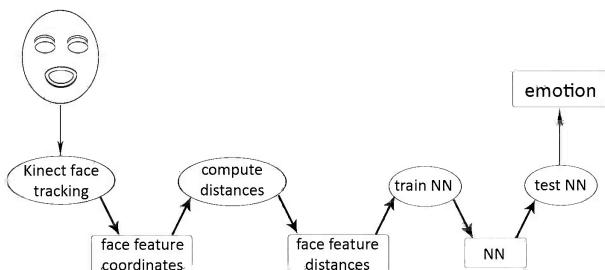


Fig. 4 System structure

Table 1 Distances measured on a sample face (in mm)

Distance	Happiness	Sadness	Fear	Anger	Surprise	Disgust
UpperBrowEyeCenter	14	15	16	10	17	11
LowerBrowEyeCenter	10	11	12	7	13	7
UpperBrowCornerEyeCorner	13	14	15	9	16	9
LowerBrowCornerEyeCorner	9	10	11	6	12	7

- {happiness, sadness}
- {fear, surprise}
- {anger, disgust}

We can thus train the neural network to output 3 classes, and afterwards split each class into the 2 emotions.

Nevertheless, absolute distance values differ for different face geometries or distances and angles of tracking. Therefore, we should compute relative distances, i.e. the ratio between the absolute distances for an emotion and the absolute distances for the neutral expression. Table 2 shows the values for the relative distances.

We then trained a simple feedforward neural network with one hidden layer of 10 neurons, and the logistic sigmoid (*logsig*) as activation function. We decided on this structure after training and testing the network with different parameters.

LBEC - LowerBrowEyeCenter	EW - EyeWidth	ELW - ExtLipsWidth
UBEC - UpperBrowEyeCenter	EH - EyeHeight	ILW - IntLipsWidth
LBCEC - LowerBrowCornerEyeCorner	UB - UpperBrows	ELH - ExtLipsHeight
UBCEC - UpperBrowCornerEyeCorner	LB - LowerBrows	ILH - IntLipsHeight

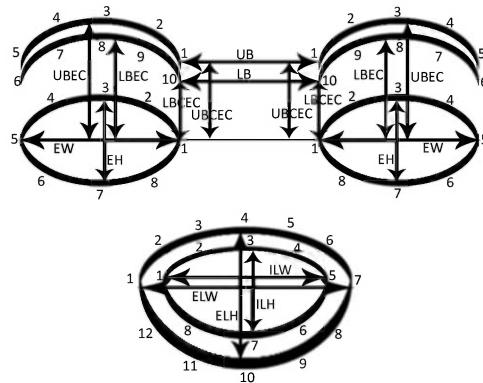
**Fig. 5** Distances measured between face features

Table 2 Relative distances (in mm)

Relative distance =	Happiness	Sadness	Fear	Anger	Surprise	Disgust
Absolute distance / Neutral						
UpperBrowEyeCenter	1.01	1.14	1.21	0.9	1.25	0.85
LowerBrowEyeCenter	1.03	1.18	1.28	0.88	1.31	0.82
UpperBrowCornerEyeCorner	1.02	1.15	1.22	0.86	1.26	0.83
LowerBrowCornerEyeCorner	1.03	1.2	1.3	0.87	1.34	0.81

4.1 Results

We tracked different persons expressing the 6 emotions. We put together all the data (except for one person), with a total of more than 1000 frames for each emotion, and split the dataset into training, validation and test data. The results were as follows:

- for absolute distances, the predictive accuracy was 87.56%; when testing the network on data not in the training set, the accuracy decreased to 78.83%
- for relative distances, the predictive accuracy was 91.41%; when testing the network on data not in the training set, the accuracy was almost the same, 91.07%

The results thus confirm that relative distances offers better results than absolute distances. The overall accuracy is greater in the case of relative distances, and also the accuracy for a new person hasn't decrease as in the case of absolute distances.

5 Conclusion

We have built an agent-based system that aims at detecting user emotions and making recommendations based on user affective state. Particularly, we presented the case of a smart museum where the agent anticipates the gallery that the visitor would enjoy most. In this paper we focused on emotion recognition from facial expressions using the Kinect sensor.

Our approach was to follow the psychological model in order to make the automatic affect recognition. We computed the distances between the face features and we showed that relative distances provide a better accuracy than absolute distances. The overall accuracy of 91% proves the feasibility of this approach.

Nevertheless, the results could still be improved. First, we should reconsider the relevance of the distances. Secondly, we should increase the dataset with more persons, to cover more face geometries or ways of expressing emotions. Thirdly, the current method doesn't consider rotations or translations of the person's head, so the tracked user must look at the sensor for accurate results.

Acknowledgements. The work has been funded by Project 264207, ERRIC-Empowering Romanian Research on Intelligent Information Technologies/FP7- REGPOT-2010-1 and by the Sectoral Operational Programme Human Resources Development 2007-2013 of the

Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/107/1.5/S/76813.

References

1. Badler, N., Smoliar, S.: Digital representations of human movement. *Computing Surveys* 11(1), 19–38 (1979)
2. Becker, E., Metsis, V., Arora, R., Vinjumur, J., Xu, Y., Makedon, F.: Smartdrawer: Rfid-based smart medicine drawer for assistive environments. In: Proceedings of the 2nd International Conference on Pervasive Technologies Related to Assistive Environments (PETRA 2009), pp. 49:1–49:8. ACM (2009)
3. Calvo, R., D'Mello, S.: Affect detection: An interdisciplinary review of models, methods, and their applications. *IEEE Transactions on Affective Computing* 1(1), 18–37 (2010)
4. Cearreta, I., López, J.M., López de Ipíña, K., Garay, N., Hernández, C., Graña, M.: A study of the state of the art of affective computing in ambient intelligence environments. In: Interacción 2007 (VIII Congreso de Interacción Persona-Ordenador, II Congreso Espanol De Informatica - CEDI 2007), pp. 333–342. Asociación Interacción Persona-Ordenador (AIPO) (2007)
5. Dellaert, F., Polzin, T., Waibel, A.: Recognizing emotion in speech. In: Proceedings of International Conference on Spoken Language Processing, pp. 1970–1973 (1996)
6. Ekman, P.: Emotions Revealed. Henry Holt and Company LLC (2003)
7. Ekman, P., Friesen, W.: Facial Action Coding System: Investigator's guide. Consulting Psychologist Press (1978)
8. El Fallah Seghrouchni, A., Olaru, A., Nguyen, N.T.T., Salomone, D.: Ao dai: Agent oriented design for ambient intelligence. In: Desai, N., Liu, A., Winikoff, M. (eds.) PRIMA 2010. LNCS, vol. 7057, pp. 259–269. Springer, Heidelberg (2012)
9. Grammenos, D., Kartakis, S., Adami, I., Stephanidis, C.: Camile: Controlling ami lights easily. In: Proceedings of the 1st International Conference on Pervasive Technologies Related to Assistive Environments (PETRA 2008), pp. 35:1–35:8. ACM (2008)
10. Hebe, N., Cohen, I., Huang, T.: Multimodal Emotion Recognition. In: Handbook of Pattern Recognition and Computer Vision. World Scientific (2005)
11. Ismail, A.-A., Florea, A.-M.: Multimodal indoor tracking of a single elder in an AAL environment. In: van Berlo, A., Hallenborg, K., Rodríguez, J.M.C., Tapia, D.I., Novais, P. (eds.) Ambient Intelligence & Software & Applications. AISC, vol. 219, pp. 137–145. Springer, Heidelberg (2013)
12. Mota, S., Picard, R.: Automated posture analysis for detecting learner's interest level. In: Proceedings in Computer Vision and Pattern Recognition Workshop, vol. 5, pp. 49–54 (2003)
13. Picard, R.: Affective Computing. MIT Press (2000)
14. Vinhas, V., Oliveira, E., Reis, L.P.: BioStories: Dynamic multimedia environments based on real-time audience emotion assessment. In: Filipe, J., Cordeiro, J. (eds.) ICEIS 2010. LNBI, vol. 73, pp. 512–525. Springer, Heidelberg (2011)
15. Yang, N., Muraleedharan, R., Kohl, J., Demirkol, I., Heinzelman, W., Sturge-Apple, M.: Speech-based emotion classification using multiclass svm with hybrid kernel and threshholding fusion. In: 2012 IEEE Workshop on Spoken Language Technology (2012), <http://www.rochester.edu/news/show.php?id=5072>

An Argumentation Framework for BDI Agents

Tudor Berariu

Abstract. This article presents a practical approach to building argumentative BDI agents. As in the last years the domain of argumentation reached maturity and offers now a very rich and well structured abstract theory, the challenge now is to put this work into practice and prove its usefulness in real applications. There is a high interest from the multi-agent systems community in applying argumentation for agents' defeasible reasoning.

The main goal of the work presented in this paper was to provide the means to enable argumentative capabilities in BDI agents. For this reason, Jason, a platform for the development of multi-agent systems using the BDI model of agency, was extended with a module for argumentation. The proposed argumentation module is decoupled from the BDI reasoning cycle as it operates only on the belief base of the agents and does not interfere in the execution of plans, creation of goals, or agent's commitments. Although no protocol for argumentation-based dialogues is proposed here, agents can engage in any such dialogues as the argumentation module makes suggestions of attacks to put forward in conversation or gives structured justifications for different beliefs. An instantiation of Dung's abstract framework is used with state of the art structure of arguments and ways of attack and defeat between arguments.

1 Introduction

One fundamental aspect in artificial intelligence and multi-agent systems is agent reasoning about the external world, itself or the actions to take at a certain point of time.

One common problem in agent logic is ensuring the consistency of beliefs after information conflicting with previous view of the world is perceived. After such

Tudor Berariu
University Politehnica of Bucharest, Bucharest, Romania
e-mail: tudor.berariu@gmail.com

an information update, the consistency of the belief set must be preserved and this process is called *non-monotonic reasoning*. Two distinct approaches to its formalization are known. First, there are different extensions to the classical logic like McDermott and Doyle's modal operator M [12], Reiter's logic for default reasoning [14] or McCarthy's circumscription theory [11]. The second class of formalisms for mechanizing non-monotonic reasoning are the various families of truth maintenance systems (TMS), first proposed by Doyle in [6].

Close to Doyle's vision, argumentation provides an alternative way to deal with non-monotonic reasoning: arguments can support existing beliefs or can act as counterarguments against them. In this way, solving conflicts between arguments does the job of belief revision.

The paper presents an argumentation-based system able to maintain consistency of the belief base of BDI agents, while hiding the functioning of the argumentation mechanisms from the BDI reasoning cycle. The system allows the agents to query the argumentation module for suggestions of attacks or argumentation-based justifications for accepted or rejected beliefs

The approach starts from Dung's abstract argumentation framework [7] but it is centered towards the integration of argumentation with the practical aspects of BDI agent behaviour by developing an argumentation module to be integrated in the Jason platform¹.

The document is structured as follows: section 2 introduces some fundamental concepts from argumentation theory, section 3 describes my practical solution to the problem stated above, while section 4 brings an example of non-monotonic reasoning that uses argumentation in the proposed system.

2 Abstract Argument Systems

The work of Dung [7] is considered the first major step towards *argumentation systems* as it provides the means to use argumentation theory for non-monotonic reasoning. Dung offers an abstraction of the attack relation between arguments, on top of which refutation, a central concept in argumentation, is built. A *refutation* of an argument is an opposed argument that attacks the original argument and defeats it. There are several ways to attack an argument: by asking an appropriate critical question that raises doubt about the acceptability of the argument, by questioning one of its premises or by putting forward counter-arguments that oppose the original argument, meaning that the conclusion of the opposing argument is the opposite (negation) of the conclusion of the original argument. There are also more complex ways to attack arguments: doubting about the relevance of the premises to the conclusions or even about the relevance of the argument in relation to the issue of the dialogue, arguing that a set of arguments commit a logical fallacy.

An example of Dung-style representation of arguments is that shown in Figure 1. In that example, argument A1 attacks A2, A3 and A4; A3 attacks A5; A5 and A6 attack each other.

¹ <http://jason.sourceforge.net/Jason/Jason.html>

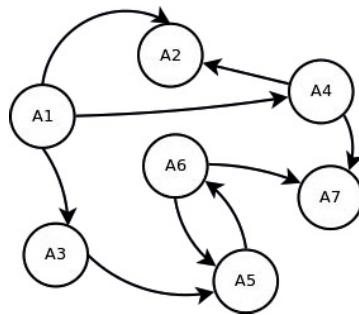


Fig. 1 Representation of argument attacks

An *abstract argument system* is a tuple $\langle \mathcal{A}, \mathcal{R} \rangle$ where \mathcal{A} is a set of *arguments* and \mathcal{R} is a binary relation over \mathcal{A} called *attack relation*.

It is clear that, while arguments attack each other, they cannot stand together and their status is subject to *evaluation*. That means that the *justification state* of each argument must be determined. An argument is regarded as *justified* if it survives the attacks it receives and it is *rejected* otherwise.

The *argument evaluation* needs a formal method that describes the steps of the process or the states of arguments based on some criteria. These formal methods are called *argumentation semantics* and there are two categories: *extension-based* and *labeling-based*.

Extension-based semantics specifies how to obtain sets of arguments \mathcal{E} where each extension E of an argumentation framework $\langle \mathcal{A}, \mathcal{R} \rangle$ is a subset of \mathcal{A} containing a set of arguments that can stand together. In extension-based semantics the *justification state* of an argument is defined in terms of membership of the respective argument to the extensions.

In extension-based semantics two alternative types of justification, namely *skeptical* and *credulous* can be considered. In a formal way, for an argumentation framework AF and a semantics \mathcal{S} , an argument a is:

- *skeptically justified* if and only if $\forall E \in \mathcal{E}_{\mathcal{S}}(AF) : a \in E$
- *credulously justified* if and only if $\exists E \in \mathcal{E}_{\mathcal{S}}(AF) : a \in E$.

Using this classification, *justification states* of arguments can be defined. An argument a is:

- *justified* if and only if $\forall E \in \mathcal{E}_{\mathcal{S}}(AF) : a \in E$ (this corresponds to skeptical justification);
- *defensible* if and only if $\exists E_1, E_2 \in \mathcal{E}_{\mathcal{S}}(AF) : a \in E_1, a \notin E_2$ (this corresponds to credulous justification)
- *overruled* if and only if $\forall E \in \mathcal{E}_{\mathcal{S}}(AF) : a \notin E$ (arguments that cannot be justified are rejected).

Building on definitions for acceptable arguments (that are defended against all attacks) and admissible sets (that contain only acceptable arguments), Dung proposes four *traditional* semantics:

complete E is a complete extension if and only if E is admissible and every argument of \mathcal{A} which is acceptable wrt. E belongs to E

grounded The grounded semantics are easier to explain by the process of building them incrementally from the unattacked arguments. The arguments attacked by them can be suppressed. The process is repeated until no new arguments arise after a deletion step. The set of all initial arguments identified so far is the grounded extension.

stable A stable extension attacks all arguments not included in it.

preferred An extension E is a *preferred extension* of AF if E is as large as possible and able to defend itself from attacks.

There are other extension-based semantics proposed in the literature: *stage semantics* [15] (the stage extension is the maximum conflict-free set); *semi-stable semantics* [4] (the semi-stable extension is the maximal complete extension); *ideal semantics* [8]; *CF2 semantics* [1]. Another semantics proposed, the *prudent semantics* [5] is based on a more extensive notion of attack in the context of traditional semantics: an argument a *indirectly attacks* an argument b if there is an odd-length attack path from a to b . Recent work on argumentation semantics explore methods of local computation of extensions [10].

This represents the theoretical basis for our work.

3 Building Argumentative Agents in Jason

There are some attempts in the literature to combine the BDI model of agency with argumentation based reasoning. Most of these works concern negotiation or other dialogue games where agents have to respect a certain protocol. The challenge tackled in this document is a bit different. The goal here is to build an argumentation module attached to the BDI engine that enables general argumentation capabilities to the agents, not just for a specific dialogue game. By this claim I mean that an agent capable of a non-monotonic argumentation based reasoning can participate into argumentation dialogues if the rules to follow a specific protocol are programmed in the agent and, possibly, an argumentation strategy is defined. For persuasive agents, such a strategy should work using a proper argumentation semantics that identifies the set of arguments to defeat, as the one proposed in [9]. An aspect one should be careful when designing an argumentation based reasoning engine for a BDI agent is that information comes from multiple sources which can influence the status of each piece of information from an argumentation point of view and, as a consequence, the entire reasoning process.

The framework used in this project is based on the latest instantiations of Dung's abstract formalism, especially on Prakken's work [13]. As in [13], I used an abstract argumentation framework with structured arguments and three types of attacks

between them: rebutting, undercutting and undermining. Also, the framework takes advantage of the distinction between contradiction and contrariness (as in [2]).

To construct multi-agent systems in the BDI paradigm, we chose Jason a platform that uses a high level language (an extension of AgentSpeak) [3] for programming the agents.

The approach taken in this work was to separate the BDI *strict* reasoning cycle (the Jason reasoning cycle) from the argumentation *defeasible* reasoning. The result is that there are two reasoning modules that operate on the agent's knowledge, but not on the same piece of information. This might seem a disadvantage at first, but there are rational reasons to do that.

One advantage of decoupling the BDI reasoning module from the argumentation module is the fact that agents might receive (sense) a lot of data that is irrelevant from an argumentative point of view. There are beliefs which generate goals and plans that do not need an argumentation treatment. For example, an agent has a motion sensor and its only use is decide whether to turn on the light or not. Representing all this data as arguments in an argumentation framework brings an unneeded overhead.

3.1 Coupling the Modules

When a new belief is added, deleted or a new set of percepts are received from the environment, before the BDI logic starts to treat the new events, these are intercepted by the argumentation module. Here a sequential update on many layers is done and a set of visible modifications (additions and removals) to the belief base is sent to the BDI reasoning engine in Jason.

First, from the set of all beliefs and percepts received (for either addition or removal), only those that are relevant for the argumentation system are kept. The others are passed untouched to the BDI engine. The filtering is done on the basis of the language and on the formulas that appear in rules. If there is no rule that has as an antecedent or as its consequent a certain formula, then the latter is not relevant for argumentation. This verification adds a small computational overhead for the cases when beliefs or percepts that are irrelevant to argumentation pass through the argumentation module.

Next, these beliefs and percepts are transformed into premises in the knowledge base of the argumentation theory. Here, the type of premise is decided as described later. There are several rules that apply in order (user expressed preferences, custom functions or default rules from the argumentation theory: e.g. the Carneades model).

On the basis of the modified premises (new premise, premises whose types were changed, deleted premises) the set of matched rules is also updated. New arguments are formed if new rules match or old arguments are deleted if they correspond to premises that were removed.

At the next level, the list of attacks between arguments is updated. First, attacks from and to deleted arguments are removed. Second, new attacks for new arguments based on the contrariness function are computed.

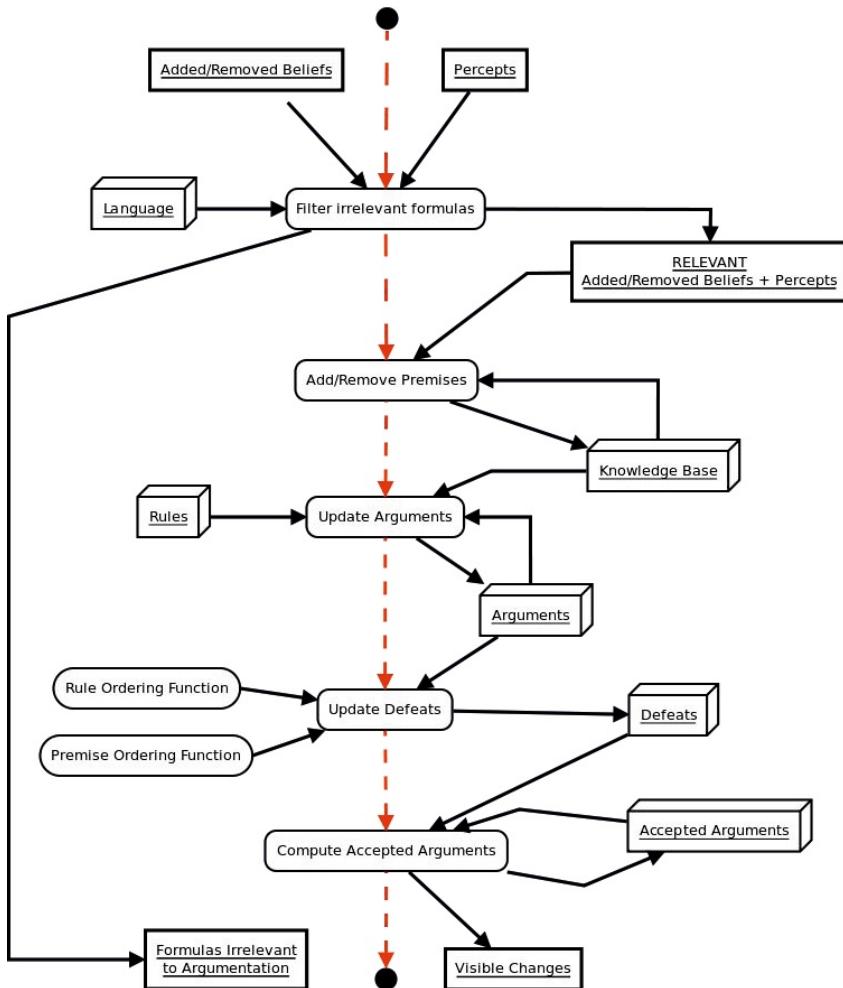


Fig. 2 The course of action in the argumentation framework when a new belief is added/removed or perceived

With the new list of attacks, the successful attacks which result in defeat are filtered. This is done using the theory described in the previous section.

The updates propagate further to the semantic extensions of arguments. The last set of accepted arguments is saved. With the new graph of defeats between arguments a new set of acceptable arguments is computed. The preferred semantics are used as they have the highest level of consistency. The differences between the old set and the new set of accepted arguments are actually the changes that are transmitted back to the Jason agent to be processed by the agent's belief revision function.

3.2 Controlling the Argumentation Module through Beliefs

As all added or removed beliefs pass through the argumentation framework, the same strategy was used to control the argumentation module with beliefs that have reserved predicate names.

In order to control the way in which conclusions are accepted, I introduced the `argument_acceptability(arg)` belief, where `arg` takes on of the following values: {`CREDULOUSLY_ACCEPTABLE` for *credulously S-acceptable* conclusions, `SKEPTICALLY_ACCEPTABLE`, or `RANDOM`} . In a similar manner, using the `plausible_argument_orderings` one can configure one of the two plausible argument orderings defined by Amgoud: `LAST_LINK` or `WEAKEST_LINK`.

There are two special beliefs for defining strict and defeasible rules.

`defeasible_rule(RuleName, RuleText)` adds a defeasible rule to the argumentation system, while `strict_rule(RuleName, RuleText)` adds a strict rule. Rules are given in the following format:

$$\textit{literal}_1 \& \dots \& \textit{literal}_n \Rightarrow \varphi$$

where φ is the conclusion and $\textit{literal}_i$ with $i \in \{1, \dots, n\}$ are antecedents.

Next, there are two predicates used to define the contradiction and the contrarieness relations: `contradictory(Literal1, Literal2)` and `contrary(Literal1, Literal2)`. As undercutting attacks against the appliance of a rule are needed, beliefs in the form `evidence_against(Literal, RuleName)` add such information in the argumentation system.

In order to differentiate between different types of premises based on their source, the `premise_from(AgentName, PremiseType)` beliefs can be used.

As the argumentation module should help the agent in persuasion, inquiry, negotiation or deliberation dialogues, another belief that queries the argumentation module has been added. `why(Proposition)` interrogates the argumentation module to find out why the respective argument is accepted or rejected. The response comes in a belief `because(X, Y)` where X is one of `{in, out}` and possible answers are:

- `because(out, unknown)` : which means that the proposition in the query is not in the knowledge base. Agent is not aware of the query formula or its negation.
- `because(in, premise(premise_type))` : which means that the specific proposition was added as a premise in the knowledge base and is not the result of any form of reasoning (strict or defeasible). The current status of that premise is `premise_type(axiom, assumption, ...)`.
- `because(out, ¬Proposition)` which means that the formula in the query is not itself in the knowledge base, but its negation is currently an accepted argument.
- `because(in, ¬Proposition)` which means that the formula in the query is not itself in the knowledge base, but its negation is currently an overruled argument.

- `because(in, Rule)` : which tells the agent that the argument corresponding to the proposition in the query is currently accepted and it is the result of applying the (defeasible or strict) rule Rule.
- `because(out,ListOfDefeats)` : which returns a list with all the conclusions of the arguments that defeated the current one.

The argumentation engine maintains a graph-like representation for the arguments and their justifications. In order to resolve `why(Proposition)` queries, the arguments space is explored starting from the belief matching Proposition.

4 Case Study: Business Trip

In what follows a scenario for argumentation implemented in Jason is described. This example uses a single agent for whom the argumentation module works just as a nonmonotonic reasoning agent.

In the next figures, a green box represents an accepted argument, a red box a refuted one, while blue is for strict rules and yellow for defeasible rules.

Consider an agent that has only one belief that represents the information that is the birthday of one of his friends and one defeasible rule which says that if it is the birthday of a friend, then he probably goes to a party. Consider now that we add another belief to the agent. Suppose he has a meeting in Berlin, so he will probably fly to Berlin. He cannot be both in Berlin and Bucharest at the same time, so the two propositions are marked as contradictory.

```
friend_s_birthday.
defeasible_rule("DR1", "friend_s_birthday => go_to_party").
meeting_in_berlin.
defeasible_rule("DR2", "meeting_in_berlin => fly_to_berlin").
contradictory("go_to_party", "fly_to_berlin").
```

Now, arguments for `go_to_party` and `fly_to_berlin` attack each other. As there are no preferences between defeasible rules or premises that can be applied here, both attacks are successful.

There are two preferred extensions, one that contains `fly_to_berlin` and one that contains `go_to_party`. Hence, if the agent is credulous, he will accept both and if he is a skeptical agent, he will accept none of the two (see Figure 3). Both provide good information about the uncertainty of arguments, but neither is useful from a practical point of view. An agent cannot use all the arguments that he credulously accepted, as there can be pairs of conflicting arguments and it's not useful to reason just on the arguments that can be skeptically accepted as that could lead to no action taken (not going to the party and not flying to Berlin).

Now, let's consider, that in general, if you have to go to a business meeting you go in most of the cases. But you are not going to all of your friends' birthday parties. So, we add a rule that says that the appliance of rule DR2 is more probable than the appliance of rule DR1.

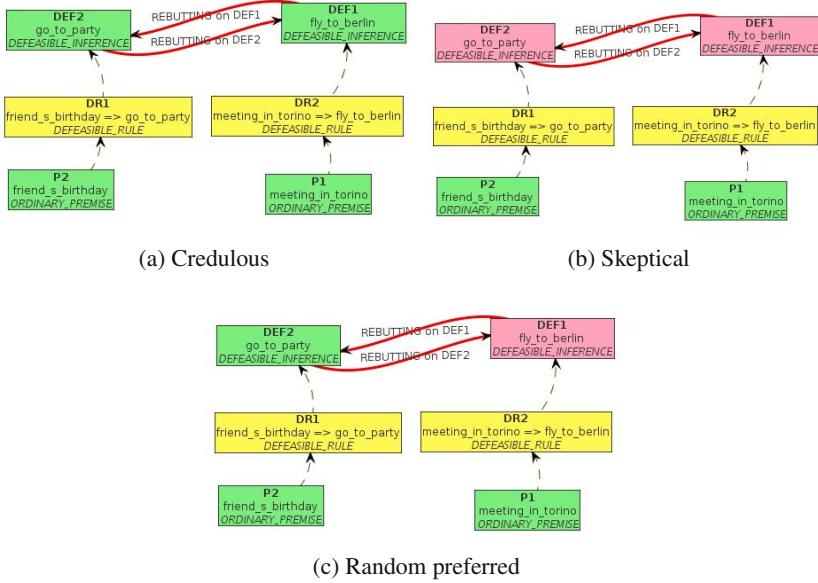


Fig. 3 Accepted arguments depending on the acceptability principle

```
argument_ordering("LAST_LINK").
prefer_rule("DR1", "DR2").
```

Now, as just the attack from DEF1 to DEF2 succeeded, there is only one preferred extension (so all arguments are both skeptically and credulously accepted). The new state of the arguments is represented in Figure 4.

Now, suppose that the agent watched the news were the possibility of another eruption of the volcano in Island was announced. This would mean that the airports will be closed. This last argument provides a situation in which the rule `meeting_in_berlin => fly_to_berlin` cannot be applied. That results in an undercutting attack as in Figure 5.

The Jason code for the information added is:

```
volcano_at_news.
evidence_against("airport_closed", "DR2").
strict_rule("SR1", "volcano -> airport_closed").
defeasible_rule("DR3", "volcano_at_news => volcano").
```

As the argument corresponding to `airport_closed` is not attacked by any other argument, it will defeat the argument with the conclusion `fly_to_berlin` in all preferred extensions.

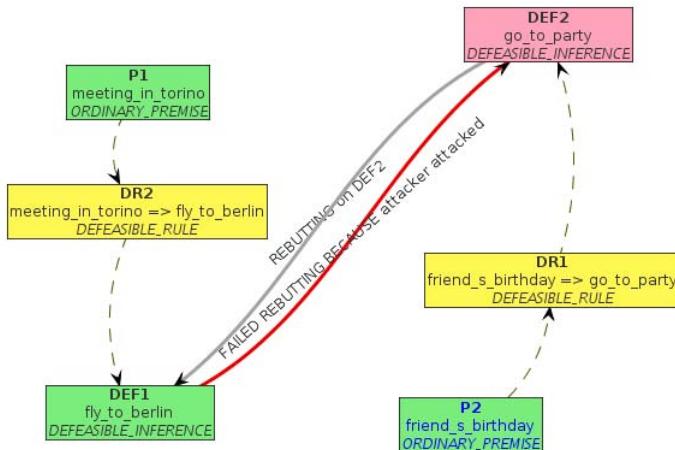


Fig. 4 Argument ordering using rule preferences

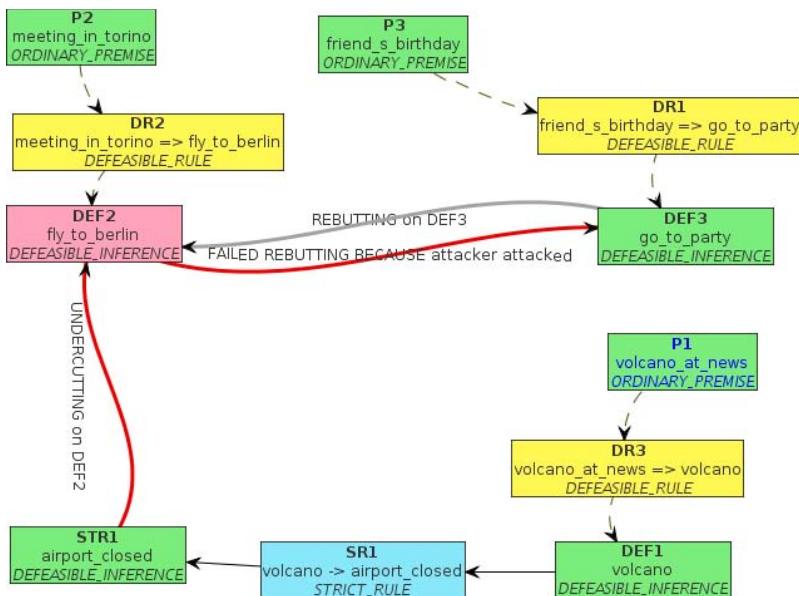


Fig. 5 Example of undercutting attack

5 Conclusions and Future Work

This paper presents a framework for supporting abstract argumentation reasoning in BDI agents. State-of-the art concepts from argumentation theory were put in

practice using a popular platform for developing BDI agents, Jason. In our work, we implemented a module to be used when programmers want to add argumentation reasoning to their agents built in Jason. The use of this module does not affect the rest of the functioning of Jason as the module intercepts any addition or removal of beliefs in forms of percepts , messages from other agents or mental notes of the agent itself and outputs to the Jason reasoning engine the effects of applying argumentation to that specific change. What the agent will be aware of (in the Jason context) is a set of consistent beliefs, as the other pieces of information corresponding to defeated arguments are hidden.

Future work will investigate the impact argumentation has on different types of dialogues between agents, especially in negotiations. Another line of development we foresee emerging from this work is the study of the utility of different semantics in practical scenarios of multi-agent interaction.

Acknowledgment. This work has been funded by project ERRIC (Empowering Romanian Research on Intelligent Information Technologies), number 264207/FP7-REGPOT-2010-1.

References

- [1] Baroni, P., Giacomin, M., Guida, G.: SCC-recursiveness: a general schema for argumentation semantics. *Artificial Intelligence* 168(1-2), 162–210 (2005)
- [2] Bondarenko, A., Dung, P.M., Kowalski, R.A., Toni, F.: An abstract, argumentation-theoretic approach to default reasoning. *Artificial intelligence* 93(1), 63–101 (1997)
- [3] Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak using Jason. John Wiley and Sons, Ltd. (2007)
- [4] Caminada, M.: Semi-stable semantics. In: Computational Models of Argument: Proceedings of COMMA, pp. 121–130 (2006)
- [5] Coste-Marquis, S., Devred, C., Marquis, P.: Prudent semantics for argumentation frameworks. In: 17th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2005, pp. 5–572. IEEE (2005)
- [6] Doyle, J.: A truth maintenance system*. 1. *Artificial Intelligence* 12(3), 231–272 (1979)
- [7] Dung, P.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games*. 1. *Artificial Intelligence* 77(2), 321–357 (1995)
- [8] Dung, P., Mancarella, P., Toni, F.: A dialectic procedure for sceptical, assumption-based argumentation. In: Proceeding of the 2006 conference on Computational Models of Argument: Proceedings of COMMA 2006, pp. 145–156. IOS Press (2006)
- [9] Gratie, C., Florea, A.M.: Argumentation semantics for agents. In: Cossentino, M., Kaisers, M., Tuyls, K., Weiss, G. (eds.) EUMAS 2011. LNCS, vol. 7541, pp. 129–144. Springer, Heidelberg (2012)
- [10] Gratie, C., Florea, A.M., Meyer, J.J.C.: General directionality and the local behavior of argumentation semantics. In: Ossowski, S., Toni, F., Vouros, G.A. (eds.) Proceedings of the First International Conference on Agreement Technologies, AT 2012, Dubrovnik, Croatia, October 15-16. CEUR Workshop Proceedings, vol. 918, pp. 113–127. CEUR-WS.org (2012)
- [11] McCarthy, J.: Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence* 13(1-2), 27–39 (1980)

- [12] McDermott, D., Doyle, J.: Nonmonotonic logic 1. *Artificial Intelligence* 13, 41–72 (1980)
- [13] Prakken, H.: An abstract framework for argumentation with structured arguments. *Argument & Computation* 1(2), 93–124 (2010)
- [14] Reiter, R.: A logic for default reasoning. *Artificial Intelligence* 13(1-2), 81–132 (1980)
- [15] Verheij, B.: Two approaches to dialectical argumentation: admissible sets and argumentation stages. In: Computational Dialectics Workshop, pp. 3–7. Citeseer (June 1996)

Using Emotion as Motivation in the Newtonian Emotion System

Valentin Lungu, Andra Băltoiu, and Șerban Radu

Abstract. The main goal of this work has been to develop an emotion simulation model and agent architecture that provides artificial characters in virtual environments with believable behavior in order to enhance virtual environment experiences for human users; this means that emotions should act as motivation for agent behavior, influence and improve perception, reasoning and decision-making.

We put forth the Newtonian Emotion System, a representation model vectorial in nature, easily integrated into computational environments and firmly grounded in Plutchik's theory of emotion. The system functions according to psychological theory, influencing the way that a character perceives the environment. The system does not influence reasoning. However it does influence the agent's actions through the **motivation mechanism**, which we consider vital for the agent to have believable affective states.

1 Introduction

The main goal of our work is to provide virtual characters with the ability of emotion expression. Several attempts have been made to endow artificial agents with an emotional layer [10, 9, 14, 2, 3]. However, we believe that in order for artificial agents to have deep, meaningful and believable emotions, they need their emotional layer to serve a similar function to that of its natural human counterpart.

Emotion in humans has been strongly linked to motivation. Emotions are a part of the human evolutionary legacy [1, 11, 13, 4, 12], serving adaptive ends, acting as a heuristic in time-critical decision-processes (such as fight-or-flight situations). We believe that emotions act as a subsystem that enhances human behavior, by stepping up brain activity in arousing circumstances, directing attention and behavior,

Valentin Lungu · Andra Băltoiu · Șerban Radu

University POLITEHNICA of Bucharest, Bucharest, Romania

e-mail: valentin.lungu@cs.pub.ro

<http://aimas.cs.pub.ro/people/valentin.lungu/>

establishing importance of events and act as motivation. The Newtonian Emotion System [5, 6, 8, 7] that we developed will influence agent behavior by establishing the importance of events and by influencing knowledge processing, as well as provide the agent with an emotional state that it will be able to express and that will further influence its behavior.

2 The Newtonian Emotion System

This chapter presents our new improved emotion simulation technique based off of Plutchik's emotion taxonomy. However, the emotion representation scheme has been streamlined, the emotion feature being reduced to a four-dimensional vector, and emotion interactions follow Newtonian interaction laws that are easier to learn and understand. The architecture has also been reduced to the bare minimum necessary for emotion simulation and expression while still influencing character behavior and memory in the same way. The new architecture allows for any behavior generation technique (belief-desire-intention, expert systems, behavior trees, finite state machines) and any machine learning algorithm (SVM, linear regression, clustering), in essence, allowing designers to develop a system that suits them best, while incorporating emotion simulation.

2.1 Newtonian Emotion System Space

In this section we introduce the Newtonian Emotion Space, where we define concepts that allow emotional states to interact with each other and external factors, as well as the two laws that govern the interaction between an emotional influence and an emotional state. The Newtonian Emotion Space is based on the work of R. Plutchik, shown in Fig. 1.

Laws of Emotion Dynamics

The following are two laws that form the basis of emotion dynamics, to be used in order to explain and investigate the variance of emotional states within the emotional space. They describe the relationship between the forces acting on an emotional state and its motion due to those forces.

Theorem 1. *The velocity of an emotional state remains constant unless it is acted upon by an external force.*

Theorem 2. *The acceleration \mathbf{a} of a body is parallel and directly proportional to the net force \mathbf{F} and inversely proportional to the mass m : $\mathbf{F} = m \cdot \mathbf{a}$*

Emotion Center and Gravity

The emotion space has a center, the agent's neutral state, a point in space to which all of the agent's emotional states tend to gravitate (usually $(0,0,0,0)$, however, different characters might be predisposed to certain kinds of emotions, thus, we should be

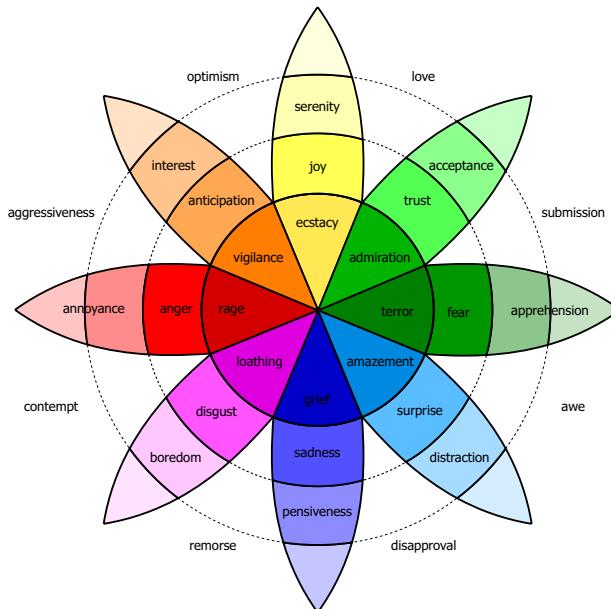


Fig. 1 Newtonian emotion space

able to specify a different emotional centre, and instill a certain disposition, for each character; we also use different emotion state mass to show how easy/hard it is to change an agent's mood). In order to represent this tendency we use gravitational force:

$$\mathbf{G} = m \cdot \frac{\mathbf{p} - \mathbf{c}}{||\mathbf{p} - \mathbf{c}||} \cdot k_g,$$

where p is the current position, c is the center and k_g is a gravitational constant. This force ensures that, unless acted upon by other external forces, the emotional state will decay towards the emotion space center.

2.2 Newtonian Emotion System Architecture

We developed an emotion subsystem architecture (Fig. 2) that interposes itself between the agent's behavior module and the environment. The subsystem models the process described by Lazarus.

Events perceived from the environment are first processed by the appraisal module, where an emotional force is associated with it. The resulting list of events is then sorted in descending order according to magnitude and fed into the agent's perception module. This is done in accordance with the psychological theory of attention narrowing, so that in a limited attention span scenario, the agent will be aware of the more meaningful events.

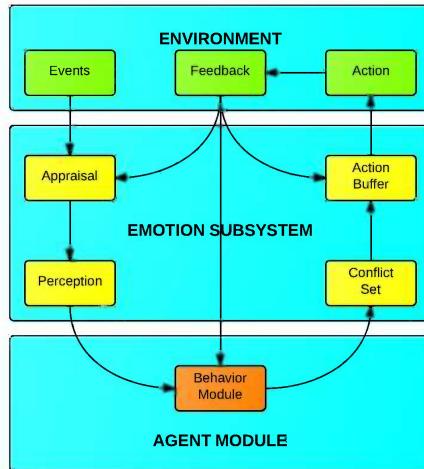


Fig. 2 Emotion system architecture

We treat the agent behavior module as a black box that may house any behavior generation technique (rule based expert system, behavior trees, etc.). The interface receives as input events perceived from the environment and produces a set of possible actions. The conflict set module takes this set of rules and evaluates them in order to attach an emotional state to each. Out of these, the action whose emotion vector most closely matches the agent's current state is selected to be carried out.

$$\arg \min_{i \in \{conflictset\}} \arccos \frac{\mathbf{e}_{agent} \cdot \mathbf{e}_i}{\|\mathbf{e}_{agent}\| \cdot \|\mathbf{e}_i\|}$$

Feedback from the environment has an associated emotion force vector that actually affects the agent's emotional state. Feedback is distributed to the appraisal and conflict set modules as well as the agent's behavior module. Actions undertaken are temporarily stored in an action buffer for feedback purposes.

2.3 Test Scenario

Feedback from the environment has an associated emotion force vector that affects the agent's state. The agent attempts to maximize its Gain (joy and truth axes) and minimize its Risk (fear and surprise axes), and thus will choose actions that increase one while decreasing the other; however, the agent does not know what the feedback will be. It is the learning's module task to attempt to predict it, based on the current context. The better the prediction, the better informed a decision the agent can make.

We chose to use linear regression to predict feedback in our application. Linear regression is a way of modelling the relationship between a dependent variable and one or more explanatory variables. The data is modelled using a linear combination

of the set of coefficients and explanatory variables. Linear regression is a supervised machine learning technique that estimates unknown model parameters from the data.

In our case, the algorithm attempts to learn the feedback method explained. The dependent variable is the emotional force received by the agent, while the explanatory variable is a feature vector constructed based on the action's context (Fig. 3). The structure of this array is made up of three segments:

- **Base.** Contains the basic information about the action.
- **Agents.** Contains information about the agents involved in the action.
- **Objects.** Contains information about the objects (other than equipment, i.e. containers, traps) involved in the action.

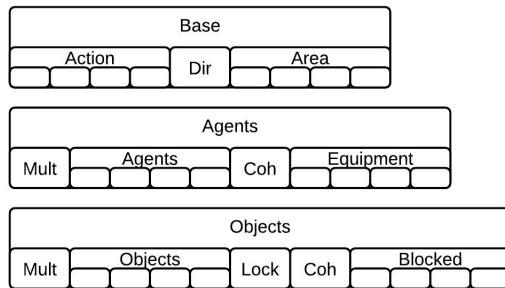


Fig. 3 Action context

The algorithm then receives the correct feedback from the environment which modifies the weights in the algorithm according to the difference between the estimated feedback force and the actual feedback force.

2.4 Learning in the Newtonian Emotion System

Both appraisal (percept and action appraisal) modules use machine learning techniques in order to learn to predict the outcome of events and actions, respectively. As there are many viable options for which learning technique to use (hidden markov models, neural networks, q-learning, POMDP), we use a plug-and-play model where the technique can be replaced [7]. The appraisal module attempts to predict the emotional feedback received from the environment based on characteristics of the event to classify and previous feedback. The goal of the appraisal module is to better label (and thus establish priority of) events for the perception model. The conflict set module works in a similar way based on the characteristics of actions taken. The goal of the conflict set module is to settle conflicts between competing actions by selecting the most appropriate action to be performed. The action that has the greatest Gain-Risk ratio will be chosen. Both modules treat the event,

rule respectively, configuration as the observable state of the model, and attempt to predict the next hidden state (the emotional feedback force). Let us take the following example: character A's behavior module generates as the next action, an attack on character B. The following simplified feature vector for the action is built, that encodes information about the current context (feelings about the action, feelings about our target and action direction). From previous experience, the agent knows that the action would raise it's aggression level (feelings about the attack action are on the anger and anticipation axes [0,0,-.9,-.3]); we also assume that we are somewhat afraid of the character that we are attacking [0,0,.7,0], and we know that we are performing the action, so action direction is 1. Based on this feature vector, the appraisal module predicts an emotional feedback with a dominant on the anger and anticipation axes of [0,0,-.2,-.2]. We assume this action has a satisfactory Gain/Risk ratio and gets selected and executed. We also assume that it succeeds. According to the environment feedback formula we used, the result would be [0,0,-.2,-.3] which means that the estimation was close, but could be better. The feedback result is added to the learning module data set, and the results are nudged in the direction of $[0,0,-.2,-.3] - [0,0,-.2,-.2] = [0,0,0,-.1]$, for a better future prediction. The perception appraisal module works in a similar way to appraise perceived actions, however the percepts are sorted according to a factor based on their distance to the agent's current emotional state (minimum) or their distance to the average emotional state of events perceived this turn (maximum). This is so that an agent will prioritize events that it resonates with or that stand out from the norm.

2.5 Personality Filter

The personality filter allows an agent's designers to skew the agent's perception of events. It defines a perception in interpreting emotional forces when applied to feedback received from the environment. The personal filter consists of a four dimensional vector (each element matching an axis of the Newtonian Emotion representation scheme) with values between -1 and 1 that scales the agent's emotional feedback according to personal criteria. For example, an agent may feel fear more accurately than joy and as such, its personal filter should reflect this, having a lower scaling factor for joy than for fear. Another agent may be extremely slow to anger, this would be reflected in the personal filter as a very low scaling factor for the anger axis. It would even be possible to change the sign altogether so that some agents may feel anger instead of fear or joy instead of sadness (leaving room for the representation of psychologically damaged individuals).

3 Emotion as Motivation

The emotional feedback that an agent receives from the environment influences its behavior. This is achieved by selecting an action from the conflict set provided by the behavior model according to the feedback that the agent predicts from the environment.

In order to evaluate its effect on the agent's current state, we put forth the notion of tensor product between two states, because of the information this gives about the influence that one axis has on another. This product has an interpretation as stress produced by an emotional force on the current emotional state, showing how the force is, relative to the current state of the agent. This is called the emotional impact and the tensor product is computed between the two forces acting on the agent: the Gravity force, pulling the agent towards its center, and the learned action force (estimated feedback).

$$\begin{aligned} \text{Stress} &= [\text{Feedback}] \cdot [\text{Gravity}] \\ &= \begin{bmatrix} f_{\text{joy}} \\ f_{\text{trust}} \\ f_{\text{fear}} \\ f_{\text{surprise}} \end{bmatrix} \cdot [g_{\text{joy}} \ g_{\text{trust}} \ g_{\text{fear}} \ g_{\text{surprise}}] \end{aligned} \quad (1)$$

This allows the assessment of the influence that each emotion axis of the action has on those of the current state. In line with Lazarus' appraisal theory, emotional evaluation implies reasoning about the significance of the event, as well the determination of the ability to cope with the event. Therefore we define two metrics for assessing emotional experiences. Well-being measures the effect the action has on the positive axes (Joy-Trust) of the current state, thus establishing the gain implied by the action; while Danger describes the risk, as it evaluates the action in terms of its influence on the Fear-Surprise axes.

When choosing an action, an agent estimates how much its current well-being is influenced by the action in regard to how much danger the action implies, a gain-risk ratio. In terms of the tensor product, this involves a column-wise computation with respect to the Well-being axes of the agent's current state (for Gain) and a row-wise computation with respect to the Danger axes of the action (for Risk).

$$\begin{aligned} \text{Gain} &= g_{\text{joy}} \cdot (f_{\text{joy}} + f_{\text{trust}} + f_{\text{fear}} + f_{\text{surprise}}) \\ &\quad + g_{\text{trust}} \cdot (f_{\text{joy}} + f_{\text{trust}} + f_{\text{fear}} + f_{\text{surprise}}) \end{aligned} \quad (2)$$

$$\begin{aligned} \text{Risk} &= f_{\text{fear}} \cdot (g_{\text{joy}} + g_{\text{trust}} + g_{\text{fear}} + g_{\text{surprise}}) \\ &\quad + f_{\text{surprise}} \cdot (g_{\text{joy}} + g_{\text{trust}} + g_{\text{fear}} + g_{\text{surprise}}) \end{aligned} \quad (3)$$

Thus the *Impact* factor can be used to assign priorities to all actions in order to resolve the conflict set.

$$\text{Impact} = \text{Gain} - \text{Risk} \quad (4)$$

Let's assume that a character's current state is [.3,.5,-.1,-.1] (gravity = [-.5,-.8,.16,.16]). We will further assume that the agent finds itself in a conflictual situation surrounded by other agents. We will also assume that the behavior module will only generate options to attack the other characters, and that the appraisal module predicts the correct feedback. In the table, the possible feedback options are various degrees of

- **fear and surprise (alarm)** - the attack fails, but was not expected to
- **anger and surprise (outrage)** - when the attack succeeds, but was not expected to
- **anger and anticipation (aggression)** - when the attack succeeds and was expected to

According to our feedback formula, we have the following impact factors:

Table 1 Attack impact table

Gravity	Feedback	Gain	Risk	Impact
[-0.5 -0.83 0.16 0.16]	[0 0 0.2 0.3]	0.4	0.3	0.10
[-0.5 -0.83 0.16 0.16]	[0 0 -0.6 -0.6]	1.56	1.176	0.38
[-0.5 -0.83 0.16 0.16]	[0 0 -0.4 0.3]	0.13	0.098	0.03
[-0.5 -0.83 0.16 0.16]	[0 0 -1 -1]	2.6	1.96	0.64
[-0.5 -0.83 0.16 0.16]	[0 0 1 0.6]	-2.08	-1.568	-0.51
[-0.5 -0.83 0.16 0.16]	[0 0 1 0.3]	-1.69	-1.274	-0.42

In table 1) we can see that the agent will prefer the two options that will increase it's aggression, will be neutral towards attempting to attack an agent that will surprise it, and will be reluctant to attack when it might fail.

As a further example, let us assume that the agent is in the same state and that it is surrounded by objects that it can pick up, the only actions generated are actions to pick up objects and the joy felt when picking up an object is equal to $\frac{value(object)}{value(object)+wealth(agent)}$.

Table 2 Take impact table

Gravity	Feedback	Gain	Risk	Impact
[-0.5 -0.8 0.16 0.16]	[0.1 0 0 0]	-0.13	0	0.13
[-0.5 -0.8 0.16 0.16]	[0.2 0 0 0]	-0.26	0	0.26
[-0.5 -0.8 0.16 0.16]	[0.3 0 0 0]	-0.39	0	0.39
[-0.5 -0.8 0.16 0.16]	[0.4 0 0 0]	-0.52	0	0.52
[-0.5 -0.8 0.16 0.16]	[0.5 0 0 0]	-0.65	0	0.65
[-0.5 -0.8 0.16 0.16]	[0.6 0 0 0]	-0.78	0	0.78

As we can see in the second table (Fig. 2), for varying item values, we get different priorities. The agent would prefer to pick up the most valuable item first.

4 Conclusion

We have provided virtual characters with the means of artificial expression and methods for its integration into intelligent artificial agent architectures. The system is firmly grounded in theory, respecting PLutchik's taxonomy, Lazarus' appraisal

model, replicating the way emotions influence human perception and motivating a character's actions in the environment. We have developed a game artificial intelligence framework and demonstrated how easily the Plug-and-play architecture is integrated with the AI framework as well as the game engine used. We have also succeeded in keeping the system complexity down in order to make the system more accessible and easy to use and adopt. Last, but not least, the Plug-and-Play system is scalable (as the system only intervenes within the agent's context, the overhead is linear) and both the New Newtonian Emotion System and the Plug-and-Play interface are easily expandable.

5 Future Work

There are several application domains that the model can easily expand to. The first among these would be as a trust and reputation model. Emotions play such a role in humans. The process is called reciprocal altruism, an emergent norm in human systems that states that one person will help another, seemingly without any possibility of reciprocation, in the hopes that the initial actor will be helped themselves at a later date - in artificial intelligence terms, this means that an agent would act in a manner that temporarily reduces its fitness while increasing another agent's fitness, with the expectation that the other agent will act in a similar manner at a later time. Emotions serve as an evaluator to help spot cheaters that try to abuse the system. We could use the Trust-Disgust axis in order to quantify how a given agent performs when a contract has been agreed upon, and use the Surprise-Anticipation axis in order to evaluate how an agent conforms to emergent but non-contractual norms. In order for the system to be useful in spotting *cheaters*. This can be achieved in a centralized or distributed manner, depending on goals.

Acknowledgements. The work has been funded by Project 264207, ERRIC-Empowering Romanian Research on Intelligent Information Technologies/FP7-REGPOT-2010-1.

References

- [1] Heuer, F., Burke, A., Reisberg, D.: Remembering emotional events. *Memory and Cognition* 20, 277–290 (1992)
- [2] Clore, G., Ortony, A., Collins, A.: The cognitive structure of emotions. Cambridge University Press, Cambridge (1988)
- [3] Pereira, D., Oliveira, E., Moreira, N.: Formal modelling of emotions in BDI agents. In: Sadri, F., Satoh, K. (eds.) CLIMA VIII 2007. LNCS (LNAI), vol. 5056, pp. 62–81. Springer, Heidelberg (2008)
- [4] Kensinger, E.A.: Remembering emotional experiences: The contribution of valence and arousal. *Reviews in the Neurosciences* 15, 241–251 (2004)
- [5] Lungu, V.: Rule-based system for emotional decision-making agents. In: Sisteme Distribuite, Suceava (2009) ISSN 2067–5259

- [6] Lungu, V.: Artificial emotion simulation model. In: 7th Workshop on Agents for Complex Systems (ACSYS 2010) held in conjunction with 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2010, Timisoara (2010)
- [7] Lungu, V.: Newtonian emotion system. In: Proceedings of the 6th International Symposium on Intelligent Distributed Computing - IDC 2012. Springer Series on Intelligent Distributed Computing, vol. VI, pp. 307–315 (September 2012)
- [8] Lungu, V.: Artificial emotion simulation model and agent architecture. Advances in Intelligent Control Systems and Computer Science 187, 207–221 (2013)
- [9] Sabouret, N., Ochs, M., Corruble, V.: Simulation of the dynamics of non-player characters' emotions and social relations in games. IEEE Transactions on Computational Intelligence and AI in Games 1(4), 281–297 (2009)
- [10] Ioerger, T.R., El-Nasr, M.S., Yen, J.: Peteei: a pet with evolving emotional intelligence. In: Proceedings of the third annual conference on Autonomous Agents, Seattle, Washington, United States, pp. 9–15 (April 1999)
- [11] Schachter, S., Singer, J.: Cognitive, social, and physiological determinants of emotional state. Psychological Review 69, 379–399 (1962)
- [12] Scherer, K.: What are emotions and how can they be measured? Social Science Information 44(4), 695–729 (2005)
- [13] Sharot, T., Phelps, E.A.: How arousal modulates memory: Disentangling the effects of attention and retention. Cognitive, Affective and Behavioral Neuroscience 4, 294–306 (2004)
- [14] Strauss, M.: Eric: a generic rule-based framework for an affective embodied commentary agent. In: AAMAS 2008: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 97–104 (2008)

Distributed Reputation Mechanism Using Semantic Repositories

Andreea Urzica and Ileana Bobric

Abstract. Trust and reputation represent topical subjects in computer science, given the evolution of multi-agent systems, e-commerce or recommender systems. The present paper proposes a distributed reputation mechanism and analyses the users' satisfaction degree in order to compare various methods of computing reputation. A scenario has been chosen as a use-case for the proposed mechanism, scenario that consists of an open real estate market. The mechanism may be further applied to any offer-request market and implementation may be used as a framework for competitively testing various aggregation functions.

1 Introduction

Reputation mechanisms bring a significant benefit especially to the new entrants within an open, heterogeneous multi-agent system. This paper proposes a reputation mechanism based on aggregating the opinions available in various semantic repositories and shows how the entities within virtual communities are able to identify and use the reputation levels of others.

The present work also provides some methods for aggregating reputation information and discusses their comparison on the same set of data. Other decision methods can be further added and used with the proposed system. The testing framework offers enough realism and has a modular architecture. Different sets of preferences and differently altered perception (even when confronted with the same reality) are simulated for each agent.

The paper is structured as follows: Section 2 briefly discusses similar efforts in building and identifying reputation within multi-agent systems, Section 3 describes the dynamics of the proposed mechanism and Section 4 introduces several methods

Andreea Urzica · Ileana Bobric

University Politehnica of Bucharest, 313 Spl. Independentei, Bucharest, Romania
e-mail: andreea.urzica@cs.pub.ro, ileana.bobric@cti.pub.ro

that can be used by this mechanism for computing reputation. In Section 5 we present some experimental results and Section 6 concludes.

2 Related Work

Trust and reputation management is a topical issue in Internet-based application and there are many research efforts ongoing (e.g. [2], [3], [5], [7]). Similarly to REGRET [5], the proposed model also uses direct experience and third party information, but allows the agent to decouple the methods of combining the available information and computing reputation from the opinion storage and representation means. By contrast to the model proposed in [7], our model has the ability of invoking any other method with no costs of re-defining the structure of the model. Concerning the applicability of the domain tackled, in [6] trust and reputation management methods are used for semantic web service selection and ranking solution based on QoS. In contrast to classical reputation mechanism (e.g. [1]), we propose a different approach, that allows agents to compute subjectively the reputation, making use of all the evidence (opinions of other users) recorded in the semantic repositories.

An early attempt towards a computational view on trust in multi-agent systems is Stephen Marsh's thesis [4] that takes inspiration from sociological research on trust. The proposed model includes an implicit integration with the three facets identified by Marsh: basic, general and situational trust. In our model the situation is clearly described, right from the start, by the query addressed to the opinion repository. The general trust can be updated by each agent dynamically with regard to the degree of accuracy of the authors that it can perceive.

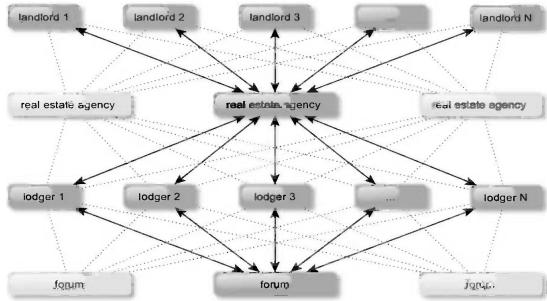
3 Real Estate Market Decisions Based on Reputation

The problem tackled here is a frequent one: the request-offer matchmaking on a free market. This problem may be instantiated to many application domains, such as commercial transactions, job fairs, transport and tourism facilities etc. The domain chosen to be studied in this paper is the real estate market, in particular, the house renting interactions. By selecting and aggregating the existing opinions of a group concerning a house, the potential lodgers are able to compute, in a decentralised manner, the reputation built by the landlord of that house.

The system includes two types of communication media for the agents: a real estate agency and a forum. The real estate agency intermediates the interaction between lodgers and landlords. It mainly selects for the lodger all the offers corresponding to their requirements. The forum allows lodgers to communicate with other lodgers, so they can express their opinions concerning the quality of their interaction with the landlords. The forum also provides filtering criteria. Figure 1 provides an overview on the system, along with the message exchange paths.

Upon querying an agency with regard to certain requirements about a house (e.g. location, dimension, price, etc.), an agent may be presented multiple choices. The role of the reputation mechanism is to help the agent make a decision and select the

Fig. 1 Interactions between the agents within the system



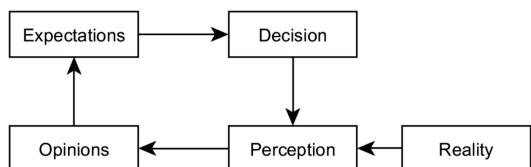
option that will bring her maximum utility. Upon using the real estate property for a while, a lodger is able to issue an opinion and post it on a forum, thus enriching the real estate knowledge, for the use of other potential lodgers. This cycle is illustrated in Figure 2. In order to evaluate the performance of the reputation mechanism during the simulation the following conventions have been made:

- There is a certain *real* value of each real estate property, known only by the simulation designer, called *reality* hereafter.
- The opinions posted by agents on forums *accurately* reflect their perceptions concerning the houses.
- All agents may have an *altered perception* of the reality. Each agent perceives the value of the house altered by a certain percent, according to a certain scheme. The alteration percent is specific to every agent and it does not vary over time.
- The agents may be more objective or more blinded in perceiving the reality, depending on the *bias percent*. The closer this percent is to zero, the more accurate is the perception of the agent. This is how we can simulate the difference in opinion of different agents concerning the same object. The aim of the reputation model is to facilitate the collaboration between agents with similar views, since there is no universal good or bad, it all depends on the evaluation system.

4 Computing the Reputation

Four different decision methods are provided for illustrating the functioning of the reputation mechanism. An opinion consists of values assigned, by an agent, to various attributes describing the analysed object (i.e the apartment). For each of the

Fig. 2 The life cycle Expectations–Opinions



methods, the chosen house will be the one with the maximum value for the expectations function.

The first aggregation method provided as an example in this paper is named *Expert Opinion*. An agent using this method will make its decisions based on the opinion of a single author on the forum considered by the agent to be an expert with regard to a certain real estate property (maybe because it has the greatest number of posts concerning that property). For each property returned as an option complying with its request, an agent computes the expectations as an average function on the scores of all the attributes of the house.

The second method provided as an example in this paper is called *The Weakest Link*. By using this decision method the agents choose a house by considering that the most relevant rate received by a house to be the minimum one. For each house matching the requirements, for each opinion concerning the house, all the rates for each criterion are aggregated into one single opinion rate. The aggregation function used may be a weighed sum, where the weights reflect the relevance of the criterion to the agent.

The decision method called *Dynamic Trust* aggregates all the opinions of all users found on a forum concerning a house. In addition this method makes use of the trust factor associated to each author as reputation information provider. An agent using this method will iterate through the list of opinions for each house and, for each opinion issuer will aggregate its opinions and weight them by the trust factor associated to that issuer. The expectation level built by an agent about a house x is computed using the following formula.

$$\text{expectations}(x) = \frac{\sum_{i=1}^m \text{AuthorAverage}(\text{Author}_i)}{\sum_{i=1}^m \text{Trust}(\text{Author}_i)}, \quad (1)$$

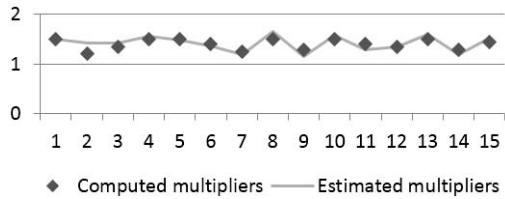
After using the house for a given period of time, the agent is able to compute the difference between its current *perception* and the *expectations* it had. The higher the similarity, the higher is the trust. For instance, if Alice's perception about a house is 9, and she knows that Bob's opinion about the same house is 6, she may learn that a future opinion of Bob should be multiplied by 1.5 in order to be useful for Alice. $\alpha \in [0, 1]$. The graphic in Figure 3 shows how this mechanism provides a good estimation of the adjustment.

$$\text{MyOpinion} = 9; \text{AuthorAggOpinion}(\text{Bob}) = 6; \text{Trust}_i(\text{Bob}) = 1.5 \quad (2)$$

$$\text{Trust}_i(y) = \text{Trust}_i(y) + \alpha \times (\text{Trust}_{i+1}(y) - \text{Trust}_i(y)) \quad (3)$$

The fourth method is called *Personal Experience*. This method allows an agent to grant a higher importance to its own opinions concerning the object to be appraised if previous experience concerning the object exists. This method balances the opinions of the community against the agent's personal opinions according to the amount of personal experience. For each of the options between which an agent has to decide upon, the agent computes the personal experience factor. The personalExperience-Factor will give a higher weight to its own opinions if the agent has more opinions concerning a certain landlord and will increase the weight for the opinion of others

Fig. 3 Opinion bias prediction



if the agent knows rather little about the same landlord. The personal experience factor concerning a house x is determined by extracting the following information from a forum:

$$\pi(x) = \text{personalExperienceFactor}(x) \quad (4)$$

$$\pi(x) = \frac{\text{number of personal opinions about } x}{\text{total number of opinions concerning } x} \quad (5)$$

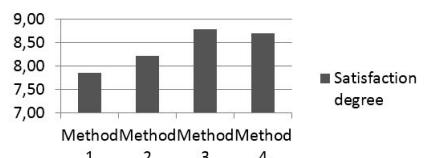
$$\text{expectations}(x) = \pi \times f(\text{personalOpinions}(x)) + (1 - \pi) \times f(\text{authorOpinions}(x)) \quad (6)$$

5 Implementation and Experimental Results

The proposed reputation model aims at promoting the service providers that offer the best facilities and discourage those with lower quality service. A success marker for the reputation model would be the higher preference of the agents within the community towards high quality service providers than towards lower quality providers. The experimental results have shown the correlation between the lodgers' preferences and landlords' quality of service.

The four reputation extraction methods have been tested comparatively by measuring the satisfaction degree for each agent. The satisfaction value is inversely proportional to the disappointment value. The disappointment value is the difference between the perception of the object and the expectations level. The bars in Figure 4 show the average satisfaction degree of the agents corresponding to each of the four reputation extraction methods compared. We can conclude that the reputation extraction method best identifying the behaviour of the service providers is *Dynamic trust*. Thus, a trust evaluating mechanism for the reputation information providers brings an important benefit to the reputation model.

Fig. 4 Agent satisfaction degree based on the reputation extraction method used



6 Conclusions

This paper proposes a decentralised reputation extraction mechanism that can be further applied to any offer-request market. The implementation may be used as a framework for testing competitively various aggregation functions.

The proposed mechanisms shows good results when applied to open, heterogeneous multi-agent systems. A significant aspect introduced by the present paper consists in analysing the *opinion formation lifecycle* and highlighting the factors that may influence each step.

The model includes several reputation extraction methods and a comparative study on the results. The simulation design captures realistic agent behaviour, distinguishing agent perception from the objective reality, and allowing agents to develop different perceptions of the same reality, according to their own internal structure.

Acknowledgements. The work presented in this paper has been funded by Project 264207, ERRIC-Empowering Romanian Research on Intelligent Information Technologies / FP7-REGPOT-2010-1.

References

1. Bilgin, S., Singh, M.P.: A DAML-Based Repository for QoS-Aware Semantic Web Service Selection. In: Proceedings of the IEEE International Conference on Web Services, pp. 368–375 (2004)
2. Despotovic, Z., Aberer, K.: Possibilities for Managing Trust in P2P Networks. EPFL Technical Report No. IC200484 (2004)
3. Josang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decision Support Systems 43, 618–644 (2007), doi:10.1016/j.dss.2005.05.019
4. Marsh, S.: Formalising trust as a computational concept. PhD Thesis, University of Stirling (1994)
5. Sabater, J., Sierra, C.: REGRET: A reputation model for gregarious societies. In: Proceedings of the Fifth International Conference on Autonomous Agents, pp. 475–482 (2001)
6. Vu, L.-H., Hauswirth, M., Aberer, K.: QoS-based service selection and ranking with trust and reputation management. In: Meersman, R., Tari, Z. (eds.) OTM 2005. LNCS, vol. 3760, pp. 466–483. Springer, Heidelberg (2005)
7. Yu, B., Singh, M.P.: Distributed Reputation Management for Electronic Commerce. In: Computational Intelligence, pp. 535–549 (2002)

Author Index

- Aisopos, Fotis 249
Allende, Héctor 17
Allende-Cid, Héctor 17
Amato, Alba 261
Amato, Flora 281, 289

Băltoiu, Andra 355
Barbareschi, Mario 289
Barhamgi, Mahmoud 175
Benea, Marius-Tudor 323
Berariu, Tudor 343
Bernard, Yvonne 189
Bobric, Ileana 365
Braubach, Lars 199
Brisan, Cornel 233
Butka, Peter 119

Camacho, David 157
Carchiolo, Vincenza 67
Casola, Valentina 289
Castelli, Gabriella 93
Chatalic, Philippe 103
Chytil, Martin 5
Ciupu, Andrei 217
Coelho, Jorge 141
Copie, Adrian 271
Coppolino, Luigi 249

D'Antonio, Salvatore 249
de Amorim Fonseca, Andre 103
Di Martino, Beniamino 243, 261
Dostal, Martin 37

Fayn, Jocelyne 175
Ficco, Massimo 243

Florea, Adina-Magda 301, 335
Fortiș, Teodor-Florin 271

Gonzalez-Pardo, Antonio 157
Gusev, Marjan 77

Holubová, Irena 5
Hong, Kirak 131

Iuhasz, Gabriel 309

Jander, Kai 199
Ježek, Karel 37
Juszkiewicz, Łukasz 223

Klejnowski, Lukas 189

Longheu, Alessandro 67
Lungu, Valentin 355

Malgeri, Michele 67
Mamei, Marco 93
Mangioni, Giuseppe 67
Mazzeo, Antonino 281, 289
Mermet, Bruno 181
Miura, Takao 23
Mocanu, Irina 301, 335
Monge, Raúl 17
Moraga, Claudio 17
Moscato, Vincenzo 281
Müller-Schloer, Christian 189
Munteanu, Ligia 233
Munteanu, Victor Ion 271, 309
Muscalagiu, Ionel 163

- Nageba, Ebrahim 175
Nečaský, Martin 5
Negreanu, Lorina 211, 301
Negru, Viorel 163, 309
Nguyen, Filip 55, 87
Nguyen, Ngoc Thanh 1
Niemann, Sebastian 189
Nogueira, Luís 141
Nykl, Michal 37

Ottenwälder, Beate 131

Pentiuc, Stefan-Gheorghe 217
Picariello, Antonio 281
Pitner, Tomáš 55, 87
Pócs, Jozef 119
Pócsová, Jana 119
Pokahr, Alexander 199
Polák, Marek 5
Popa, Horia Emil 163
Popovici, Matei 211
Potuzak, Tomas 151
Puică, Mihaela-Alexandra 335

Radu, Ţerban 355
Ramachandran, Umakishore 131
Ristov, Sasko 77

Romano, Luigi 249
Rosaci, Domenico 31, 45
Rosi, Alberto 93

Sarné, Giuseppe M.L. 31, 45
Scialdone, Marco 261
Shvartsman, Inna 317
Simon, Gaele 181

Tasquier, Luca 243
Taveter, Kuldar 317
Toporkov, Victor 109
Toporkova, Anna 109
Tovarňák, Daniel 55, 87
Tselishchev, Alexey 109
Tserpes, Konstantinos 249

Urzica, Andreea 365

Vasiu, Razvan-Vlad 233
Venticinque, Salvatore 261
Vultur, Oana Mihaela 217

Yamaguchi, Makoto 23
Yemelyanov, Dmitry 109

Zambonelli, Franco 93