2012 Imperial College Computing Student Workshop

ICCSW'12, September 27–28, 2012, London, United Kingdom

Edited by

Andrew V. Jones

Editor

Andrew V. Jones
Verification of Autonomous Systems Group
Department of Computing
Imperial College London
180 Queen's Gate, London, SW7 2AZ
United Kingdom
andrewj@doc.ic.ac.uk

ACM Classification 1998
A.0. Conference Proceedings

ISBN 978-3-939897-48-4

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at http://www.dagstuhl.de/dagpub/978-3-939897-48-4.

Publication date
November, 2012

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Digital Object Identifier: 10.4230/OASIcs.ICCSW.2012.i

ISBN 978-3-939897-48-4 **ISSN 2190-6807** **http://www.dagstuhl.de/oasics**

OASIcs – OpenAccess Series in Informatics

OASIcs aims at a suitable publication venue to publish peer-reviewed collections of papers emerging from a scientific event. OASIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 2190-6807

www.dagstuhl.de/oasics

Contents

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
ICCSW

Preface

This volume contains the proceedings of the second Imperial College Computing Student Workshop (ICCSW'12). The workshop took place on 27th–28th September 2012 in London, UK and was hosted by Imperial College London.

ICCSW is an event organised with the "by students, for students" ethos in mind. All of the organisation of the workshop was done by the steering committee of Ph.D. students at Imperial College London, with all papers and reviews also being written by students. This year, ICCSW also introduced an "ambassador" programme for students who wished to act as external publicity chairs. The ambassadors proved to be invaluable as they helped increase the national and international visibility of the event.

These proceedings contain 24 original contributions in various fields from across computer science, including both theoretical and applied papers. The workshop received 47 submissions, and after rigorous peer review, the final selection was cut down to 24 papers (and even those 24 were difficult to schedule!). From ICCSW'11, this is a 105% increase in the number of submissions and a 41% increase in the number of accepted papers. Along with having more submissions, there was also a significant improvement in the quality of the papers – this made the committee's job in selecting the final programme very tough!

Both days of the workshop featured a keynote talk on a facet of computer science – our renewed thanks go out to our keynote speakers. The talks were titled:

- *Cloud Centric Networking*, by Greg Page (Cisco Systems, Inc.); and
- *The Art of Programming Language Design: Confessions of a Connoisseur*, by Gilad Bracha (Google Inc.)

ICCSW'12 once again proved to be an international event, attracting both submissions and attendees from the UK and the rest of Europe. The workshop was pleased to host in excess of 100 participants from the following countries: Cyprus, France, Germany, Italy, Poland, Romania, Switzerland and the United Kingdom. Furthermore, the workshop also received submissions from: Greece, Ireland, the Netherlands, Saudi Arabia, Slovakia and Taiwan!

We wish to thank all authors, accepted or not, who acted as reviewers, plus the handful of external reviewers. Furthermore, our thanks also go out to our sponsors: Imperial College London, who provided us with more than just financial support; Google and Cisco for their gold-level sponsorship; and, finally, HP for providing travel bursaries to numerous authors. Without the continual confidence and financial support of these organisations, ICCSW'12 would not have been possible. For this support we are truly grateful!

October, 2012
London

Andrew V. Jones

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
ICCSW

Conference Organisation

Programme Committee

Feryal Mehraban Pour Behbahani	Imperial College London
Marcel Chris Guenther	Imperial College London
Petr Hosek	Imperial College London
Andrew Vaughan Jones	Imperial College London
Roman Kolcun	Imperial College London
Nicholas Ng	Imperial College London
Maria Nika	Imperial College London
Iryna Tsimashenka	Imperial College London
Calin-Rares Turliuc	Imperial College London

Ambassadors

Shaswar Baban	King's College London
Helen Bolke-Hermanns	RWTH Aachen
Alessandra Debenedictis	George Mason University
Hans Decker	TU Dortmund
Marco Diciolla	University of Oxford
Matthew Forshaw	Newcastle Universtiy
Alex Ginsca	Iasi University
Evgenios Hadjisoterio	University of Cyprus
Jesus Omana Iglesias	Trinity College Dublin
Konstantinos Kloudas	University of Rennes
Michał Knapik	Polish Academy of Sciences
Alexandru Matei	University College London
Artur Męski	University of Łódź
Qais Noorshams	Karlsruhe Institute of Technology
Oliver Perks	University of Warwick
Hubert Plociniczak	University of Lausanne
Alireza Pourranjbar	University of Edinburgh
Sören Preibusch	University of Cambridge
Philipp Reinecke	Freie Universität Berlin
Luke Riley	University of Liverpool
Ivan Shcherbakov	Technische Universität Kaiserslautern
Son Tran	City University London
Max Tschaikowski	Ludwig Maximilian University of Munich

External Reviewers

Kamarul Abdul-Basit	Newcastle University
Lama Al Khuzayem	Imperial College London

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
ICCSW

Abeer Al-Humaimeedy	King's College London
Theofanis Apostolopoulos	King's College London
Duangtida Athakravi	Imperial College London
Michaela Bacikova	Technical University of Košice
Emanuela Boros	Alexandru Ioan Cuza University
Kleopatra Chatziprimou	King's College London
Tib Chis	Imperial College London
Gabriele Cocco	University of Pisa
William Denman	University of Cambridge
Antoine Desmet	Imperial College London
Dionysios Efstathiou	King's College London
Xiuyi Fan	Imperial College London
Valentina Fedorova	Royal Holloway, University of London
Daniele Filaretti	Imperial College London
Alexandru Lucian Ginsca	Alexandru Ioan Cuza University
Christos Hadjinikolis	King's College London
Peter Ivancak	Technical University of Košice
Martin Jantsch	Imperial College London
Gareth Jones	Imperial College London
Dominik Lakatos	Technical University of Košice
Martyn Lloyd-Kelly	University of Liverpool
Daniel Lorencik	Technical University of Košice
Michal Malohlava	Charles University, Prague
Artur Meski	Polish Academy of Sciences
Rabih Mohsen	Imperial College London
Davide Morelli	University of Pisa
Antonis Mouhtaropoulos	University of Warwick
Milan Nosal	Technical University of Košice
Kalia Orphanou	University of Cyprus
Jan-Peter Ostberg	University of Stuttgart
Mert Ozkaya	City University London
Martin Pala	Technical University of Košice
Oliver Perks	University of Warwick
Alan Perotti	University of Turin
Hubert Plociniczak	École Polytechnique Fédérale de Lausanne
Luke Riley	University of Liverpool
Marcelo Serrano Zanetti	ETH Zurich
Lei Shi	University of Warwick
Laura Surcel	University of Craiova
Wilson Tan	University of Warwick
Daniel Telgen	University of Applied Sciences Utrecht
Martin Varga	Technical University of Košice
Naveneetha Vasudevan	King's College London
Maria Vircikova	Technical University of Košice
Andrada Voinitchi	King's College London
Wenlong Wang	Imperial College London
Hu Xu	University of Dundee
Jian Zhang	University of Dundee

Supporters and Sponsors

Supporting Scientific Institutions

Imperial College London
http://www.imperial.ac.uk/

Sponsors

Google Inc.
http://www.google.com/

Cisco Systems, Inc.
http://www.cisco.com/

Hewlett-Packard Company
http://www.hp.com/

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
ICCSW

Knowledge Transformation using a Hypergraph Data Model

Lama Al Khuzayem[1] and Peter McBrien[1]

1 **Imperial College London**
South Kensington Campus
London SW7 2AZ
l.al-khuzayem11@imperial.ac.uk
p.mcbrien@imperial.ac.uk

Abstract

In the Semantic Web, knowledge integration is frequently performed between heterogeneous knowledge bases. Such knowledge integration often requires the schema expressed in one knowledge modelling language be translated into an equivalent schema in another knowledge modelling language. This paper defines how schemas expressed in OWL-DL (the Web Ontology Language using Description Logic) can be translated into equivalent schemas in the Hypergraph Data Model (HDM). The HDM is used in the AutoMed data integration (DI) system. It allows constraints found in data modelling languages to be represented by a small set of primitive constraint operators. By mapping into the AutoMed HDM language, we are then able to further map the OWL-DL schemas into any of the existing modelling languages supported by AutoMed. We show how previously defined transformation rules between relational and HDM schemas, and our newly defined rules between OWL-DL and HDM schemas, can be composed to give a bidirectional mapping between OWL-DL and relational schemas through the use of the both-as-view approach in AutoMed.

1998 ACM Subject Classification H.2.5 Heterogeneous Databases

Keywords and phrases Knowledge Transformation, Hypergraph Data Model, BAV Mappings

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.1

1 Introduction

One of the crucial impediments that hinder the realisation of the Semantic Web vision is the integration of ontologies [1, 2]. Since ontologies are a form of knowledge representation, we use the terms ontology integration (OI) and knowledge integration (KI) interchangeably.

The increasing number of ontologies that were made publicly available on the Web, has evolved the Web into a global ontology [3]. The main purpose of this global ontology is to provide a unified query interface for the local ontologies. A crucial problem in this context is how to specify the mappings between the global ontology and the local ontologies [1]. The main mapping approaches cited in the literature are Global-As-View (GAV) [4], Local-As-View (LAV) [4], Global-Local-As-View (GLAV) [5], and Both-As-View (BAV) [6].

The problem of OI has been extensively investigated in the literature (e.g. [1, 2, 7, 8, 9, 10, 11]). By closely examining these OI proposals, we have identified two things. Firstly, while BAV is the most expressive mapping approach, none have used it. In contrast to GAV, LAV, and GLAV, BAV is not only capable of providing a complete mapping between schemas in both directions, but also the mappings between schemas are described as a pathway of primitive transformation steps applied in sequence in the form of add, delete, rename,

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 1–7

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

extend, and contract. Hence a further advantage of the approach, is that composition of data mappings may be performed such that mapping two schemas to one common schema will produce a bidirectional mapping between the original two data sources [12]. Secondly, current approaches integrate ontologies represented, for example, in the Resource Description Framework Schema (RDFS) [13] or the Web Ontology Language (OWL) [14] by choosing one of them as the Common Knowledge Model (CKM) and converting all the other modelling languages into that CKM. Using a high-level CKM such as RDFS or OWL greatly complicates the mapping process. This is because there is rarely a simple correspondence between their modelling constructs [15].

In this paper, we show how to integrate knowledge bases, represented in OWL-DL, using a low-level Hypergraph Data Model (HDM) as the CKM. Our approach has the advantage of clearly separating the modelling of data structure from the modelling of constraints on the data. Moreover, the HDM supports a very small set of low-level elemental modelling primitives (nodes, edges, and constraints) which makes it better suited for use as a CKM than higher-level modelling languages [15]. The HDM is the common data model of the AutoMed DI system [12]. The AutoMed system [12] is distinguished from other DI systems for handling a wide range of data modelling languages through representing their constraints as BAV transformations [16]. Furthermore, by mapping into AutoMed's HDM language, we are then able to map the OWL-DL schemas into any of the existing modelling languages supported by AutoMed.

The remainder of this paper is structured as follows. Section 2 gives a brief description about the HDM. In Section 3, we show some of the representations of OWL-DL axioms in HDM and in Section 4, we show how previously defined transformation rules between relational and HDM schemas [16], and our newly defined rules between OWL-DL and HDM schemas, can be composed to give a mapping between relational and OWL-DL schemas. Finally, we state our conclusions in Section 5.

2 HDM Overview

In this Section, we provide a brief overview over the HDM and we refer the reader to [16] for full details. An HDM schema is a structure in which data may be held and is defined as follows:

▶ **Definition 1. HDM Schema** Given a set of $Names$ that we may use for modelling the real world, an HDM schema, S, is a triple $Nodes, Edges, Cons$ where:

- $Nodes \subseteq \{\langle\langle \mathsf{n_n} \rangle\rangle \mid n_n \in Names\}$ $Nodes$ is a set of nodes in the graph, each denoted by its name enclosed in double chevron marks.
- $Schemes = Nodes \cup Edges$
- $Edges \subseteq \{\langle\langle \mathsf{n_e}, \mathsf{s_1}, \ldots, \mathsf{s_n} \rangle\rangle \mid n_e \in Names \cup \{\_\} \wedge s_1 \in Schemes \wedge \ldots \wedge s_n \in Schemes\}$ $Edges$ is a set of edges in the graph where each edge is denoted by its name, together with the list of nodes/edges that the edge connects, enclosed in double chevron marks.
- $Cons \subseteq \{c(s_1, \ldots, s_n) \mid c \in Funcs \wedge s_1 \in Schemes \wedge \ldots \wedge s_n \in Schemes\}$ $Cons$ is a set of boolean-valued functions (constraints) whose variables are members of $Schemes$ and where the set of functions $Funcs$ forms the HDM constraint language. In this paper we only use the following:
  1. $inclusion(s_1, s_2) \equiv s_1 \subseteq s_2$
  2. $mandatory(s_1, \ldots, s_m, s) \equiv < s_1, \ldots, s_m > \rhd s$
  3. $unique(s_1, \ldots, s_m, s) \equiv < s_1, \ldots, s_m > \lhd s$
  4. $reflexive(s_1, s) \equiv s_1 \overset{\text{id}}{\rightarrow} s$

▶ Example 1. We list in here the contents of an example HDM schema that we shall later, in Figure 2, show to be equivalent to a relational schema. Note how the names of edges are sometimes given as the character '_' representing an unnamed edge.

$Nodes = \{\langle\langle$ug$\rangle\rangle, \langle\langle$ug:ppt$\rangle\rangle, \langle\langle$student$\rangle\rangle, \langle\langle$student:name$\rangle\rangle, \langle\langle$student:sid$\rangle\rangle,$
$\langle\langle$result:grade$\rangle\rangle, \langle\langle$course$\rangle\rangle, \langle\langle$course:code$\rangle\rangle, \langle\langle$course:dept$\rangle\rangle\}$

$Edges = \{\langle\langle$_,ug,ug:ppt$\rangle\rangle, \langle\langle$_,student,student:sid$\rangle\rangle, \langle\langle$_,student,student:name$\rangle\rangle,$
$\langle\langle$result,student,course$\rangle\rangle, \langle\langle$_,$\langle\langle$result,student,course$\rangle\rangle$,result:grade$\rangle\rangle,$
$\langle\langle$_,course,course:dept$\rangle\rangle, \langle\langle$_,course,course:code$\rangle\rangle\}$

$Cons = \{\langle\langle$ug$\rangle\rangle \lhd \langle\langle$_,ug,ug:ppt$\rangle\rangle, \langle\langle$ug$\rangle\rangle \rhd \langle\langle$_,ug,ug:ppt$\rangle\rangle,$
$\langle\langle$ug:ppt$\rangle\rangle \rhd \langle\langle$_,ug,ug:ppt$\rangle\rangle, \langle\langle$ug$\rangle\rangle \subseteq \langle\langle$student$\rangle\rangle,$
$\langle\langle$student$\rangle\rangle \lhd \langle\langle$_,student,student:sid$\rangle\rangle, \langle\langle$student$\rangle\rangle \rhd \langle\langle$_,student,student:sid$\rangle\rangle,$
$\langle\langle$student:sid$\rangle\rangle \rhd \langle\langle$_,student,student:sid$\rangle\rangle, \langle\langle$student$\rangle\rangle \lhd \langle\langle$_,student,student:name$\rangle\rangle,$
$\langle\langle$student$\rangle\rangle \rhd \langle\langle$_,student,student:name$\rangle\rangle, \langle\langle$student$\rangle\rangle \xrightarrow{\text{id}} \langle\langle$_,student,student:name$\rangle\rangle,$
$\langle\langle$student:name$\rangle\rangle \rhd \langle\langle$_,student,student:name$\rangle\rangle,$
$\langle\langle$result:grade$\rangle\rangle \rhd \langle\langle$_,$\langle\langle$result,student,course$\rangle\rangle$,result:grade$\rangle\rangle,$
$\langle\langle$result,student,course$\rangle\rangle \lhd \langle\langle$_,$\langle\langle$result,student,course$\rangle\rangle$,result:grade$\rangle\rangle,$
$\langle\langle$course$\rangle\rangle \lhd \langle\langle$_,course,course:dept$\rangle\rangle, \langle\langle$course$\rangle\rangle \rhd \langle\langle$_,course,course:dept$\rangle\rangle,$
$\langle\langle$course:dept$\rangle\rangle \rhd \langle\langle$_,course,course:dept$\rangle\rangle, \langle\langle$course$\rangle\rangle \lhd \langle\langle$_,course,course:code$\rangle\rangle,$
$\langle\langle$course$\rangle\rangle \rhd \langle\langle$_,course,course:code$\rangle\rangle, \langle\langle$course$\rangle\rangle \xrightarrow{\text{id}} \langle\langle$_,course,course:code$\rangle\rangle,$
$\langle\langle$course:code$\rangle\rangle \rhd \langle\langle$_,course,course:code$\rangle\rangle\}$

3 Representing OWL-DL in HDM

We now discuss how OWL-DL axioms and facts may be represented in the HDM, and hence translated into other modelling languages. For conciseness, we only discuss those OWL-DL constructs listed in Table 1, which are sufficient to describe how the OWL-DL ontology depicted in Figure 1 can be translated into the HDM shown in Figure 1(b).

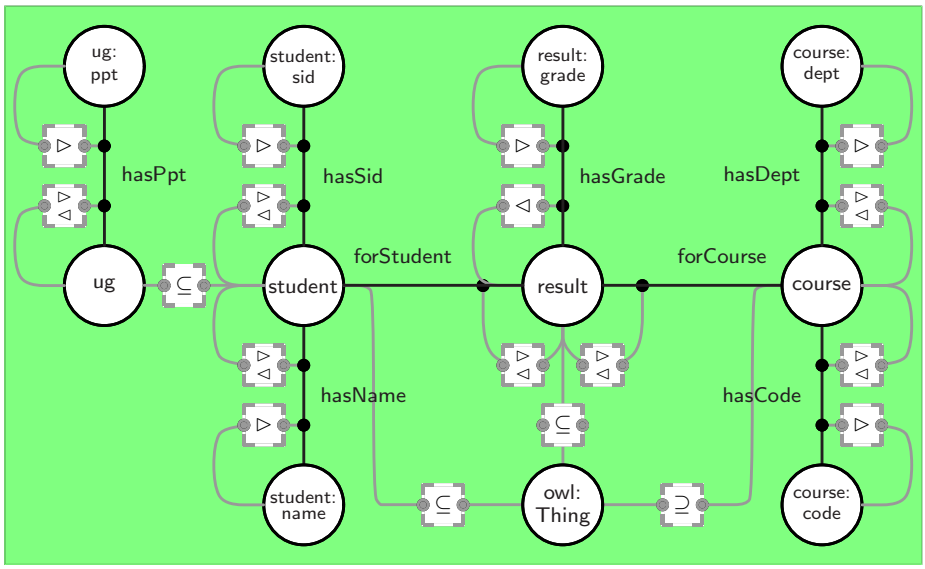
Table 1 HDM Representations of Some OWL-DL Axioms.

OWL-DL Name	DL Syntax	Scheme	HDM Representation
owl:Thing	$\top$	$\langle\langle \mathsf{owl{:}Thing} \rangle\rangle$	Node $\langle\langle \mathsf{owl{:}Thing} \rangle\rangle$
owl:Nothing	$\bot$	$\langle\langle \mathsf{owl{:}Nothing} \rangle\rangle$	Node $\langle\langle \mathsf{owl{:}Nothing} \rangle\rangle$
Class	C	$\langle\langle \mathsf{C} \rangle\rangle$	Node $\langle\langle \mathsf{C} \rangle\rangle$
SubClassOf (C_1 C_2)	$C_1 \sqsubseteq C_2$	$\langle\langle \sqsubseteq, \mathsf{C_1}, \mathsf{C_2} \rangle\rangle$	Constraint $\langle\langle \subseteq, \langle\langle \mathsf{C_1} \rangle\rangle, \langle\langle \mathsf{C_2} \rangle\rangle \rangle\rangle$
ObjectProperty	P	$\langle\langle \mathsf{P}, \mathsf{C_1}, \mathsf{C_2} \rangle\rangle$	Edge $\langle\langle \mathsf{P}, \langle\langle \mathsf{C_1} \rangle\rangle, \langle\langle \mathsf{C_2} \rangle\rangle \rangle\rangle$
FunctionalProperty	$\mathsf{T} \sqsubseteq \leq 1P$	$\langle\langle \mathsf{P}, \mathsf{C_1}, \mathsf{C_2}, \mathsf{func} \rangle\rangle$	Edge $\langle\langle \mathsf{P}, \langle\langle \mathsf{C_1} \rangle\rangle, \langle\langle \mathsf{C_2} \rangle\rangle \rangle\rangle$ Constraint $\langle\langle \rhd, \langle\langle \mathsf{C_1} \rangle\rangle, \langle\langle \mathsf{C_2} \rangle\rangle \rangle\rangle$ Constraint $\langle\langle \lhd, \langle\langle \mathsf{C_1} \rangle\rangle, \langle\langle \mathsf{C_2} \rangle\rangle \rangle\rangle$

▶ Example 2. Consider the OWL-DL schema illustrated in Figure 1(a) which represents concepts in a university Universe of Discourse and its relationships. Using the representations of OWL-DL axioms shown in Table 1, we present an equivalent HDM schema for the OWL-DL schema depicted in Figure 1(b). All classes such as owl:Thing, student, and course were represented as HDM Nodes. Functional properties such as hasName, hasSid, and hasPpt were represented as HDM edges with mandatory ($\rhd$) and unique ($\lhd$) constraints. The SubClassOf axioms such as (student $\sqsubseteq$ owl:Thing), (course $\sqsubseteq$ owl:Thing), and (ug $\sqsubseteq$ student) were represented as an inclusion constraint ($\subseteq$). Note that in the HDM diagram, HDM nodes are

student $\sqsubseteq$ owl:Thing $\sqcap$ 1 hasName.name $\sqcap$ 1 hasSid.sid
course $\sqsubseteq$ owl:Thing $\sqcap$ 1 hasCode.code $\sqcap$ 1 hasDept.dept
ug $\sqsubseteq$ student $\sqcap$ $\exists$ hasPpt.ppt
result $\sqsubseteq$ owl:Thing
result $\equiv$ 1.forStudent
result $\equiv$ 1.forCourse
1.hasGrade $\sqsubseteq$ result
grade $\sqsubseteq$ $\exists$ hasGrade
T $\sqsubseteq$ hasGrade.grade
T $\sqsubseteq$ hasGrade^{-}.result
T $\sqsubseteq$ forStudent.student
T $\sqsubseteq$ forStudent^{-}.result
T $\sqsubseteq$ forCourse.course
T $\sqsubseteq$ forCourse^{-}.result
student $\sqcap$ course $\sqsubseteq$ $\bot$

(a) An OWL-DL schema of the student-course knowledge base



(b) HDM representation of the OWL-DL schema

Figure 1 An OWL-DL schema and its equivalent HDM schema.

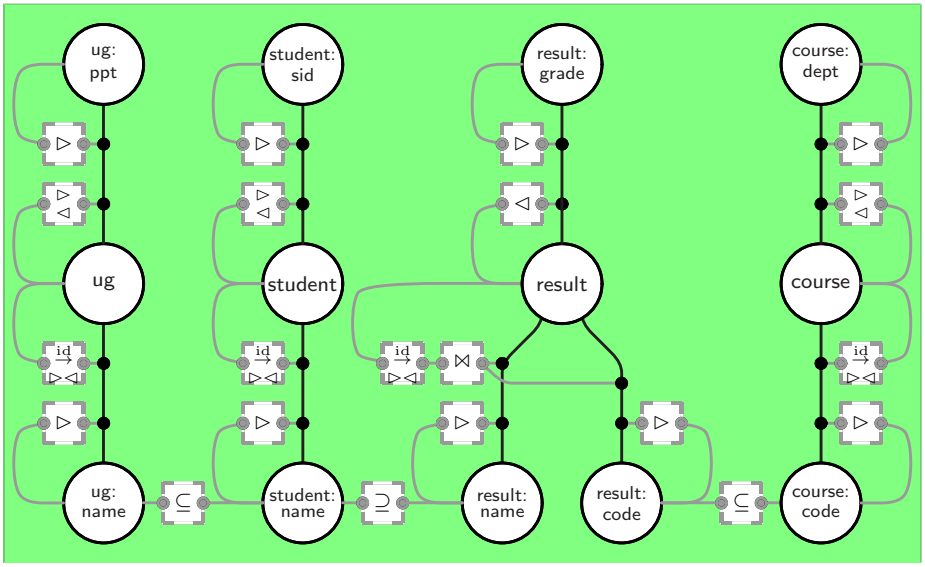
represented by white circles with thick outlines, and HDM edges are represented by thick black lines. The HDM constraint language is represented by grey dashed boxes connected by grey lines to the nodes and edges to which the constraint applies. Edges pass through black circles in a straight line, hence any edge or constraint applying to an edge meets that edge at an angle.

4 OWL-DL Knowledge Bases Transformation using the BAV Model

In [16], five general purpose equivalence mappings that allow the transformation between different modelling languages were proposed namely: Inclusion Merge, Identity Node Merge, Unique-Mandatory Redirection, Identity Edge Merge, and Node Reidentify. In this paper, we show how we can use them to transform between a knowledge model, the OWL-DL shown in Figure 1(a) and a data model, the relational shown in Figure 2(a). Taking the HDM equivalent schemas of these OWL-DL and relational schemas illustrated in Figure 1(b) and Figure 2(b) respectively and applying some of these BAV-defined mappings, we were able to transform the HDM relational schema into an HDM OWL-DL schema through 21 steps as shown below.

ug(<u>name</u>,ppt)	ug.name → student.name
student(<u>name</u>,sid)	result.name → student.name
course(<u>code</u>,dept)	result.code → course.code
result(<u>code</u>,<u>name</u>,grade?)	

(a) Relational schema for the student-course database



(b) HDM representation of the relational database schema

Figure 2 A relational schema and its equivalent HDM schema.

The first 5 steps are identical to those in transforming relational to ER HDM schemas shown in [16]. Applying these 5 steps results in Figure 3. When transforming from a key based model (such as relational) and a knowledge model that does not provide means to define keys (such as OWL-DL), we must overcome some fundamental differences which require, in our example, extending the object identifiers (OIDs) of ⟨⟨student⟩⟩, ⟨⟨course⟩⟩, and ⟨⟨result⟩⟩ respectively as illustrated in steps 6-8. The transformations associated with step 6 are illustrated in Example 3. Steps 7 and 8 are similar to step 6 thus, we do not explain them here. Step 9 is again similar to step 7 in relational and ER HDM schemas conversion given in [16]. Steps 10-13 illustrate adding the ⟨⟨owl:Thing⟩⟩ node along with three inclusion constraints (⊆) to it from the ⟨⟨student⟩⟩, ⟨⟨course⟩⟩, and ⟨⟨result⟩⟩ nodes. Finally, all we need to do to obtain the OWL-DL HDM schema is to rename the edges as shown in steps 14-21. The result of these 21 steps is the schema shown in Figure 1(b).

1. inclusion_merge (⟨⟨student:name⟩⟩,⟨⟨_,result:name, result⟩⟩)
2. inclusion_merge (⟨⟨course:code⟩⟩, ⟨⟨_, result:code, result⟩⟩)
3. identity_node_merge (⟨⟨_,ug:name, ug⟩⟩)
4. unique_mandatory_redirection (⟨⟨_, student:name, result⟩⟩, ⟨⟨_, student:name, student⟩⟩)
5. unique_mandatory_redirection (⟨⟨_, course:code, result⟩⟩, ⟨⟨_, course:code, course⟩⟩)
6. extend_OID (⟨⟨student⟩⟩ $\overset{id}{\rightarrow}$⟨⟨_,student,student:name⟩⟩)
7. extend_OID (⟨⟨course⟩⟩ $\overset{id}{\rightarrow}$⟨⟨_,course,course:code⟩⟩)
8. extend_OID (⟨⟨result⟩⟩ $\overset{id}{\rightarrow}$⟨⟨_,result,student:name⟩⟩ ⋈ ⟨⟨_,result,course:code⟩⟩)

9. move_dependants (⟨⟨student:name⟩⟩, ⟨⟨student⟩⟩, ⟨⟨_,student:name, student⟩⟩)
10. addNode (⟨⟨owl:Thing⟩⟩)
11. addCons (⟨⟨student⟩⟩ ⊆ ⟨⟨owl:Thing⟩⟩)
12. addCons (⟨⟨course⟩⟩ ⊆ ⟨⟨owl:Thing⟩⟩)
13. addCons (⟨⟨result⟩⟩ ⊆ ⟨⟨owl:Thing⟩⟩)
14. renameEdge(⟨⟨_,course,course:dept⟩⟩,⟨⟨hasDept,course,course:dept⟩⟩)
15. renameEdge(⟨⟨_,course,course:code⟩⟩,⟨⟨hasCode,course,course:code⟩⟩)
16. renameEdge(⟨⟨_,result,result:grade⟩⟩,⟨⟨hasGrade,result,result:grade⟩⟩)
17. renameEdge(⟨⟨_,result,course⟩⟩,⟨⟨forCourse,result,course⟩⟩)
18. renameEdge(⟨⟨_,result,student⟩⟩,⟨⟨forStudent,result,student⟩⟩)
19. renameEdge(⟨⟨_,student,student:sid⟩⟩,⟨⟨hasSid,student,student:sid⟩⟩)
20. renameEdge(⟨⟨_,student,student:name⟩⟩,⟨⟨hasName,student,student:name⟩⟩)
21. renameEdge(⟨⟨_,ug,ug:ppt⟩⟩,⟨⟨hasPpt,ug,ug:ppt⟩⟩)

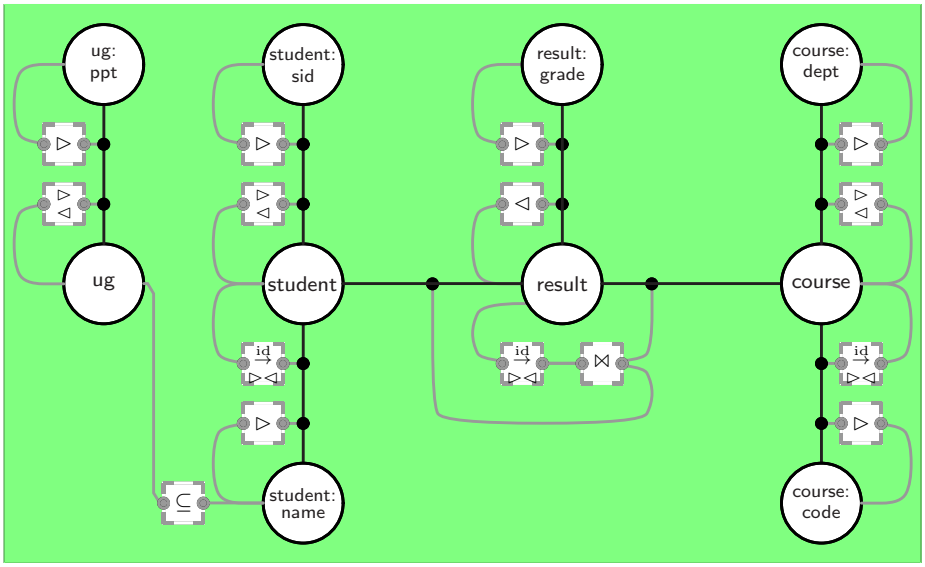


Figure 3 Intermediate HDM schema in relational to OWL-DL conversion, after steps 1–5.

▶ Example 3. Transformations associated with step 6:
1. inverse_identity_node_merge(⟨⟨student⟩⟩,⟨⟨student:oid⟩⟩)
2. deleteCons(⟨⟨student⟩⟩ $\xrightarrow{id}$⟨⟨_,student,student:oid⟩⟩)
3. node_reident(⟨⟨student⟩⟩, { ⟨x, y⟩ | ⟨o,x⟩ ∈ ⟨⟨_,student,student:oid⟩⟩∧⟨o,y⟩ ∈ ⟨⟨_,student,student:name⟩⟩})
4. deleteCons(⟨⟨student⟩⟩ $\xrightarrow{id}$⟨⟨_,student,student:name⟩⟩)
5. deleteCons(⟨⟨student⟩⟩ ◁ ⟨⟨_,student,student:name⟩⟩)
6. deleteCons(⟨⟨student⟩⟩ ▷ ⟨⟨_,student,student:name⟩⟩)
7. deleteCons(⟨⟨student:oid⟩⟩ ◁ ⟨⟨_,student,student:name⟩⟩)
8. deleteCons(⟨⟨student:oid⟩⟩ ▷ ⟨⟨_,student,student:name⟩⟩)
9. contractEdge(⟨⟨_,student,student:name⟩⟩)
10. contractNode(⟨⟨student:name⟩⟩)
11. renameNode(⟨⟨student:oid⟩⟩, ⟨⟨student:name⟩⟩)

Note that the inverse of identity node merge in transformation 1 generates a new node ⟨⟨student:oid⟩⟩, connected to ⟨⟨student⟩⟩ by a new edge ⟨⟨_,student,student:oid⟩⟩. Transformations 2-4 have the net effect of repopulating the ⟨⟨student⟩⟩ node with values of the ⟨⟨student:name⟩⟩ attribute, and deleting the keys from name and oid. Transformations 5-11 delete the ⟨⟨student:name⟩⟩ node (with its associated constraints and edge) and rename the node ⟨⟨student:oid⟩⟩ with ⟨⟨student:name⟩⟩.

5 Conclusions

In this paper, we have defined how schemas expressed in OWL-DL can be translated into equivalent schemas in HDM. We have also given an example, using the AutoMed system, that shows how to map between HDM OWL-DL schemas and HDM relational schemas which results in a bidirectional mapping between OWL-DL and relational schemas, and vice versa. Our future work will expand our approach by defining schemas expressed in other knowledge modelling languages such as OWL 2 in HDM. This might include extending the HDM constraint language in order to accomodate the richness of such modelling languages.

References

1 Calvanese, D., Giuseppe, G., and Lenzerini, M., 2001. Ontology of Integration and Integration of Ontologies. In: *DL*.

2 Noy, N., 2004. Semantic Integration: A Survey of Ontology-Based Approaches. *SIGMOD Record*, 33(4), pp. 65–70.

3 Kalfoglou, Y. and Schorlemmer, M. 2003. Ontology Mapping: The State of the Art. *The Knowledge Engineering Review*. 18(1), Publisher: Cambridge University Press, pp. 1-31.

4 Lenzerini, M. 2002. Data Integration: A Theoretical Perspective. In *Proc. PODS'02*, pp. 233-246. ACM.

5 Madhavan, J. and Halevy, A.Y. 2003. Composing Mappings Among Data Sources. In *Proc. VLDB'03*, pp. 572-583.

6 Boyd, M., Kittivoravitkul, S., Lazanitis, C., McBrien, P. and Rizopoulos, N. 2004. AutoMed: A BAV Data Integration System for Heterogeneous Data Sources. In *Proc. CAiSE'04*. LNCS.

7 Noy, N., 2003. What Do We Need for Ontology Integration on the Semantic Web. In: *ISWC'03*. Sanibel Island, Florida.

8 Udrea, O., Getoor, L., and Miller, R. 2007. Leveraging Data and Structure in Ontology Integration. In: *SIGMOD '07*. Beijing, China. pp. 449-460.

9 Lv, Y., and Xie, C. 2010. A Framework for Ontology Integration and Evaluation. *IEEE*. pp. 521-524.

10 Jimenez-Ruiz, E., Grau, B., Horrocks, I., and Berlanga, R. 2009. Ontology Integration Using Mappings: Towards Getting the Right Logical Consequences. In: *Proc. of ESWC'09*.

11 N.F. Noy and M. Musen. 2000. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proc. of AAAI'00*, Austin, TX, USA.

12 Smith, A., Rizopoulos, N., McBrien, P. 2008. AutoMed Model Management. In:*Proc. of ER'08*. LNCS, vol. 5231, pp. 542–543. Springer, Heidelberg (2008).

13 W3C. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004. Available at: http://www.w3.org/TR/rdf-schema/

14 W3C. Web Ontology Language Guide. W3C Recommendation 10 February 2004. Available at: http://www.w3.org/TR/2004/REC-owl-guide-20040210/

15 McBrien, P. and Poulovassilis, A. 1999. A Uniform Approach to Inter-Model Transformations. In: *Proc. CAiSE'99*. p.333-348, June 14-18, 1999.

16 Boyd, M. and McBrien, P. 2005. Comparing and Transforming Between Data Models via Intermediate Hypergraph Data Model. pp. 69-109, Springer-Verlag, ISBN-13 978-3-540-31001-3, ISSN 0302-9743.

A heuristic for sparse signal reconstruction

Theofanis Apostolopoulos[1]

1 **King's College, London, Department of Informatics**
Strand, London, WC2R 2LS, United Kingdom,
theofanis.apostolopoulos@kcl.ac.uk

Abstract

Compressive Sampling (CS) is a new method of signal acquisition and reconstruction from frequency data which do not follow the basic principle of the Nyquist-Shannon sampling theory. This new method allows reconstruction of the signal from substantially fewer measurements than those required by conventional sampling methods. We present and discuss a new, swarm based, technique for representing and reconstructing signals, with real values, in a noiseless environment. The method consists of finding an approximation of the l_0-norm based problem, as a combinatorial optimization problem for signal reconstruction. We also present and discuss some experimental results which compare the accuracy and the running time of our heuristic to the IHT and IRLS methods.

1998 ACM Subject Classification I.5.4 Signal Processing (Applications)

Keywords and phrases Compressive Sampling, sparse signal representation, l_0 minimisation, non-linear programming, signal recovery

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.8

1 Introduction

Over the last few years, a number of different methods for sparse approximation in signal reconstruction have arised including the Compressive Sampling technique. Compressive Sampling (CS) states that it is possible to reconstruct signals accurately and almost exactly from much fewer number of measurements than those required by the Nyquist-Shannon sampling theory. To achieve this, the method relies on two major principles: sparsity of signal and incoherence of the measurements being taken [2, 7, 8, 9, 10, 11, 13, 15, 20]. Sparsity implies that only a small percentage of the signal entries (less than 40%) in a known transform domain is nonzero or significantly different from zero [8, 11, 20, 21, 15, 20]. Incoherence in measurements states that all the collected samples of a signal are randomly generated and independent to each other [7, 8, 11, 15, 20, 21]. For simplicity, we use signals with real values each of which can be presented as a vector $X = [x_1, x_2, \ldots, x_n]$. In this article we propose a new swarm based method for sparse signal representation and reconstruction based on the key mathematical insights underlying this new theory. We compare the proposed method with two well-known signal reconstruction methods in terms of time and recovery error. The rest of this article is organised as follows: The next section presents the signal reconstruction problem and how the algorithm deals with it. Then, the algorithm is stated in Section 3, while in Section 4, we briefly describe the alternative algorithms used for comparison. Section 5 provides and presents some experimental results of our algorithm and its comparison with the other two methods.

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 8–14

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2 The Signal Reconstruction problem

Obtaining sparse solutions from an under-determined system of linear equations has been of paramount importance in the area of signal processing and analysis. The CS theory aims to obtain the sparsest possible representation of the signal $X = [x_1, x_2, \ldots, x_N]$, from an under-determined system of linear measurements $Y \in \Re^M$, so as $Y = CX$, where $X \in \Re^N$ is the signal vector we want to find and $C \in \Re^{M \times N}$ is a Sensing matrix used for under-sampling X (with $M \ll N$). This ill-posed problem can be modelled as an optimisation problem (signal reconstruction problem) as follows [2, 7, 9, 11, 13, 15, 20, 21]:

$$\min \quad \|X\|_{l_0} \quad s.t. \;\; Y = CX, \tag{1}$$

where $\|X\|_{l_0}$ is the l_0 norm which is equal to the number of non-zero components in the vector X. Finding the solution to problem (1) is NP-hard due to its nature of non-convex combinational optimization [2, 7, 9, 11, 13, 15]. For this reason many researchers suggested replacing the l_0 norm with the convex approximation of l_1 norm [7, 9, 11, 13, 15, 19]. However, it is still possible to reconstruct sparse signals using the constrained l_0-minimisation, which in many situations outperforms even l_1-minimisation in the sense that substantially fewer measurements are needed for recovery [1, 14, 16, 17, 19, 22]. The main idea is to approximate the l_0 norm by a smooth continuous function which is easier to handle and does not suffer from the discontinuities of the l_0 norm. This function can be defined as [1, 14, 16, 17, 22]:

$$\|X\|_{l_0} \approx f_\sigma(X) = N - \sum_{i=1}^{N} f_\sigma(x_i) = N - \sum_{i=1}^{N} exp(-\frac{|x_i|^2}{2\sigma^2}), \tag{2}$$

where x_i is the i-th element of the signal (vector) X of N terms (length) and $f_\sigma(X)$ is a continuous function, which belongs to the Gaussian family of functions. The σ is actually a decreasing sequence of constants $[\sigma_1, \sigma_2, \ldots, \sigma_j]$ for every iteration of the method so as to maximise the smoothed l_0 norm of the problem. Then, the problem can be defined as:

$$\max \quad f_\sigma(X) = (N - \sum_{i=1}^{N} exp(-\frac{|x_i|^2}{2\sigma^2})) \quad s.t. \;\; Y = CX \tag{3}$$

Now we have a smooth objective function, though non-linear, which is much easier for calculations. The purpose is to maximise the objective function in (3) together with the minimisation of the real parameter σ. The value of this parameter represents the tradeoff between accuracy and smoothness of the approximation.The smaller the σ, the better the approximation, while the larger the σ, the smoother the approximation. Also note that the minimisation of l_0 norm is actually equivalent with the maximisation of the f_σ for sufficiently small σ. For small values of σ, f_σ contains a lot of local maxima and thus it is difficult to maximise it. Therefore, we need to set this parameter initially very large so as to make the objective function convex and then gradually decrease it according to the value of the objective function so as to enter the region close to its global maximiser.

3 The Proposed Algorithm

In this section we present the pseudocode of the proposed method (Pseudocode (1)) together with the parameter settings used. The method is an iterative process which is based on the swarm optimisation. The computation is conducted by a group of agents, where every agent carries a solution which is slightly different from the other agents. At each iteration t the

Pseudocode 1

```
Problem: Determine a vector X s.t. CX = Y.
Inputs: σ, C, Y, Iterations, Agents, Sparsity level S, f_σ(X).
Outputs: best value f_σ*(X), best sparse vector X_*.
Proposed swarm based method:
Generate Initial X_i^(0) using (4) for every swarm i
Set σ_i^(0) = 2 × X_max for every swarm i
  While (t < Iterations) (for all iterations)
   For all Agents (for all swarms)
         Evaluate f_σ(X) for every X_i^(t)
         Find current best X_*^(t) so as max f_σ(X) and min σ
         Set X_*^(t) = X_i'^(t) (keep the best i'th solution)
         Check X_*^(t) entries for non-feasible values (Pseudocode (2))
         Consider the constrains CX = Y (project back to feasibility
        set): X_*^(t) = X_*^(t) − C^T(CC^T)^−1(CX_*^(t) − Y)
         Set all but S largest entries of X_*^(t) to zero
         Generate new solutions for all the other agents based on (5)
   End For all Agents (for all swarms)
  Set σ^(t+1) = σ^(t) × 0.5
  End While (t < Iterations)
Display the signal reconstruction error using equation (9).
```

current best solution $X_*^{(t)}$ that maximises f_σ and minimises σ is chosen. It is then corrected in terms of feasibility and bounds of its values. All the other agents are destroyed and a new solution is generated for each of them based on the previously created one. Again all the solutions are evaluated against the current best solution, which is updated, till the method completes all the number of iterations given. Also, note that the σ value is initially assigned to twice the maximum value of the vector X and then it is gradually decreased by half at each iteration. This particular assignment was chosen based on the nature of the given test vector (signal). Finally, it is notable that every solution vector generated is projected back to the feasibility set based on the constraints equation $CX = Y$ and then only the S largest entries are kept, setting all the others equal to zero. This step of the method is very important as it achieves the necessary feasibility of the new solution and also follows the sparsity level of the original vector (signal).

3.1 Initial Solution

The initial solution generated in vector format, for each swarm i, is given as:

$$X_i^{(0)} = ((C^T C)^{-1} C^T Y) + k, \tag{4}$$

where, $(C^T C)^{-1} C^T Y$ is the pseudo-inverse of matrix C, $X_i^{(0)}$ is the initial solution vector for agent i and k is a vector of small random numbers based on the lowest value of the original signal X. This k value is slightly different for every agent that carries a solution.

3.2 Solution Generation

The generation of a new solution in vector format for each swarm i is generated as:

$$X_i^{(t)} = 2 \times k^t \times X_i^{(t-1)} \times \sigma^{4L} + (1 - k^t) \times 1/M \times L, \tag{5}$$

Pseudocode 2

```
Repeat for each dimension d of vector X (for each element x_j)
        If x_j < X_min, Then x_j = X_min
        Else If x_j > X_max, Then x_j = X_max
        Else the value of entry x_j is kept the same.
End of Repeat for each dimension d of vector X
```

where, M is the number of samples, k is a vector of small random numbers between 0 and 1, different for every swarm i, t is the current iteration, while $X_i^{(t)}$ and $X_i^{(t-1)}$ is the current and the previously generated solution vector of the i-th swarm. L is the norm $\|Y - CX_*^{(t)}\|_{l_2}$ which stands for the Euclidean distance between the samples vector Y and the product between the Sampling matrix C and the current best solution at iteration t, $X_*^{(t)}$.

3.3 Solution Correction

Every solution vector $X_i^{(t)}$ created in Equation (5) is tested and corrected so as to be within the given ranges of the original vector (signal). X_{min} and X_{max} are the minimum and maximum value of the given original signal, which remain the same for all iterations. The whole procedure is presented in Pseudocode (2).

4 Alternative Algorithms

Several methods have been proposed to find the sparsest solution of the under-determined system of linear equations in (1), including many methods for obtaining signal representations in over-complete dictionaries. These methods range from general approaches, like the Basis Pursuit (BP), Orthogonal Matching Pursuit (OMP) and the method of Matching Pursuit (MP) [6, 18] to more sophisticated ones such as a Steepest Descent/Ascent methods [1, 16] together with the IHT [3, 4, 5] and IRLS [12] methods, which will be briefly described in this Section. In our point of view, all these methods have both advantages and shortcomings; some are very slow in convergence, such as BP and OMP methods, while others have low estimation quality especially for large systems of equations, such as IRLS. Furthermore, to the best of our knowledge, we are not aware of any swarm based techniques used in the Compressive Sampling framework so far.

4.1 Iterative Hard Thresholding (IHT)

Iterative Hard Thresholding Algorithm (IHT) is a simple, yet efficient, iterations based method for signal reconstruction, which uses a non-linear operator (P_k) to reduce the value of the l_0 norm at every iteration. The new solution is generated as follows [3, 4, 5]:

$$X^{(t)} = P_k(X^{(t-1)} + C^T(Y - CX^{(t-1)})), \tag{6}$$

where, Y is the samples vector, C the Sensing matrix, and $X^{(t-1)}, X^{(t)}$ are the current and the new generated solution. P_k is a hard thresholding operator that sets all but K largest elements to zero. The algorithm can be summarised in Pseudocode (3) [3, 4, 5].

4.2 Iteratively Re-weighted Least Squares (IRLS)

This algorithm tries to reconstruct sparse signals, using a re-weighted least squares method for computing local minima of the non-convex problem. It replaces the l_0 norm with a

Pseudocode 3

```
Input : Matrix C, vector Y, sparsity level k, number of iterations T
Output : Approximation vector X
The IHT Method:
Set X^(0) = 0
while (t < T) (number of iterations)
        X^(t) = P_k(X^(t-1) + C^T(Y - CX^(t-1)))
end while (t < T)
```

weighted l_2 norm, as follows [12]:

$$\min \sum_{i=1}^{N} w_i x_i^2, \quad s.t. \;\; CX = Y, \tag{7}$$

where, the weights w_i are calculated based on the previous solution so as the objective function is a first order approximation of the l_p objective function ($0 \leq p \leq 1$). The new solution at k-th iteration is generated as follows [12]:

$$x^{(k)} = Q_n C^T (C Q_n C^T)^{-1} Y, \tag{8}$$

where, Q_n is a diagonal matrix with entries $1/w_i = 1/((x_i^{(k-1)})^2 + \epsilon)^{p/2-1}$ and $\epsilon > 0$ is a small constant used to regularise the optimisation problem. The whole procedure is repeated a number of iterations based on the nature of the problem.

5 Experimental Results

In this Section we conduct numerical experiments to test the performance and the efficiency of the proposed heuristic. Table (1) shows the average time and the recovery error of the methods for the test run. It can be seen that the proposed heuristic performed faster than the others with better results. Notice that all the algorithms are based on non-linear problems and that all of them performed well in the under-sampled case of 70 samples. In experiments conducted, the Revised Simplex method (used for solving the l_1 equivalent convex problem) performed better than all the previous methods (10^{-16} error) for more than 200 samples and failed in smaller sample sizes ($70, 100, 150$ samples), where the three methods discussed achieved very good results. However, all the three methods failed to recover a signal using less than 70 samples, which appears to be the limit for efficient recovery. All the computations

Table 1 Average Time and Recovery Error for IHT, IRLS and Proposed method.

Iterations	Time	Recovery Error	Complexity	Algorithm
30	0.32335	0.0338	linear	IHT
30	0.27018	0.0653	linear	IRLS
23	0.24875	0.0281	linear	Proposed method

were performed on an Intel Core2 Duo CPU (2 GHz) with 2 GB RAM, using Matlab R2010a under MS Windows 7 Ultimate. The whole experiment took less than 2 mins. A discrete time randomly generated signal (in vector format) of 500 entries with 10% sparsity (non-zero entries) has been used for 100 test runs with 70 samples. This simple signal was constructed using the Real Gaussian model (i.e. using Standard Normal distribution) to generate real

values between 0 and 10, which constitutes a realistic model for testing the efficiency of the methods. The signal reconstruction error is defined as [7, 9, 11, 15, 20]:

$$\textit{Recovery Error} \;=\; \|X - \hat{X}\|_{l_2} / \|X\|_{l_2}, \tag{9}$$

where X and $\hat{X}$ is the original and the recovered signal, while $\|X - \hat{X}\|_{l_2}$ stands for the Euclidean distance between these two vectors. Note that the Euclidean distance of the vector $\|X\|_{l_2}$ is simply the square root of the sum of the squares of its elements. The CPU time was used as a rough estimation of time in secs, while 12 agents have been used by the proposed method, during this simulation.

6 Conclusions – Future work

In this article, an efficient heuristic for finding a sparse approximation of a signal, by solving an under-determined system of linear equations with non-linear objective function, has been proposed. It is based on maximising a smooth approximation of the l_0 norm. Although the presented heuristic has no guarantee of achieving a global minimum as does its convex l_1 analogue, the local minimum found by solving the non-convex problem in (1) typically allows for accurate and successful signal reconstruction even at much higher under-sampling rates where linear optimisation fails. Overall, the method has shown to be better in accuracy for a small number of samples and a bit faster than other alternative algorithms, without adding complexity, for the same randomly generated signal in a noiseless environment. A potential improvement of this heuristic is to re-weight the smooth l_0 norm using coefficients at every iteration; a technique that has been applied successfully to similar l_0 and l_1 norm based CS problems [10, 12, 17]. The algorithm's adaption in noisy environments constitutes another realistic improvement with much higher applicability since it is already known that the IHT and IRLS have not been extensively tested in noisy environments. Finally, potential applications of this method include the areas of signal separation, de-noising in images and signals, image sparse representation and inpainting (i.e. the process of reconstructing lost parts of images) [15, 20].

Acknowledgements The author would like to particularly thank his primary supervisor, Dr. Tomasz Radzik, for his insight and constructive comments at an earlier version of this article, and the anonymous reviewers for their valuable suggestions.

References

1 S. Ashkiani, M. Babaie-Zadeh, and C. Jutten. Error correction via smoothed l_0-norm recovery. *IEEE Statistical Signal Processing Workshop (SSP)*, pages 289–292, June 2011.

2 Richard Baraniuk. Compressive sensing. *IEEE Signal Processing Magazine*, pages 118–120, 2007.

3 Thomas Blumensath. Iterative hard thresholding: Theory and practice. Technical report, Institute for Digital Communications, Signal and Image Processing, The University of Edinburgh, February 2009.

4 Thomas Blumensath and Mike E. Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2008.

5 Thomas Blumensath and Mike E. Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, May 2008.

6 T. Tony Cai. Orthogonal matching pursuit for sparse signal recovery. *IEEE Transactions on Information Theory*, 57:1–26, 2011.

7 E. J. Candès. Compressive sampling. *Proceedings of the International Congress of Mathematicians, Madrid, Spain*, 2006.

8 E. J. Candès and J. Romberg. Sparsity and incoherence in compressive sampling. *Inverse Problems*, 23(3):969–985, June 2007.

9 E. J. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, February 2006.

10 E. J. Candès, M. Wakin, and S. Boyd. Enhancing sparsity by reweighted l_1 minimization. *Journal of Fourier Analysis and Applications*, 14(5):877–905, December 2004.

11 E. J. Candès and M. B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, March 2008.

12 Rick Chartrand and Wotao Yin. Iteratively reweighted algorithms for compressive sensing. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3869 – 3872, March 2008.

13 D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, April 2006.

14 Dongdong Ge, Xiaoye Jiang, and Yinyu Ye. A note on complexity of l_p minimisation, February 2010.

15 Stephane Mallat. *A Wavelet Tour of Signal Processing, Third Edition (The Sparse Way)*. Academic Press, 3 edition, 2009.

16 H. Mohimani, M. Babaie-Zadeh, and C. Jutten. A fast approach for overcomplete sparse decomposition based on smoothed l_0 norm. *IEEE T. Signal Processing*, 57:289–301, November 2009.

17 J.K. Pant, Lu Wu-Sheng, and A. Antoniou. Reconstruction of sparse signals by minimizing a re-weighted approximate l_0-norm in the null space of the measurement matrix. *Circuits and Systems (MWSCAS),53rd IEEE International Midwest Symposium*, pages 430–433, August 2010.

18 Scott Chen Shaobing, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20:33–61, 1998.

19 Yoav Sharon, John Wright, and Yi Ma. Computation and relaxation of conditions for equivalence between l_1 and l_0 minimization. Technical report, Coordinated Science Laboratory at University of Illinois, Urbana-Champaign, 2007.

20 Jean-Luc Starck, Fionn Murtagh, and Jalal M. Fadili. *Sparse Image and Signal Processing; Wavelets, Curvelets, Morphological Diversity*. Cambridge University Press, United Kingdom, 2010.

21 Michael Wakin. Compressed sensing, September 2009.

22 Qu Xiaobo, Cao Xue, Guo Di, Hu Changwei, and Chen Zhong. Compressed sensing mri with combined sparsifying transforms and smoothed l_0 norm minimisation. *Acoustics Speech and Signal Processing (ICASSP), IEEE International Conference*, pages 626–629, March 2010.

Predicate Invention in Inductive Logic Programming

Duangtida Athakravi, Krysia Broda, and Alessandra Russo

Department of Computing, Imperial College London, U.K.
{da407,kb,a.russo}@doc.ic.ac.uk

Abstract

The ability to recognise new concepts and incorporate them into our knowledge is an essential part of learning. From new scientific concepts to the words that are used in everyday conversation, they all must have at some point in the past, been invented and their definition defined. In this position paper, we discuss how a general framework for predicate invention could be made, by reasoning about the problem at the meta-level using an appropriate notion of top theory in inductive logic programming.

1998 ACM Subject Classification I.2.6 Learning, D.1.6 Logic Programming

Keywords and phrases Predicate invention, Inductive logic programming, Machine learning

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.15

1 Predicate Invention

In Inductive Logic Programming (ILP) a hypothesis H (a set of rules) is learned from some background knowledge B and a set of observed positive and negative examples $E = E^+ \cup E^-$, using mode declarations as bias for the syntax of the rules. The learned hypothesis should be the most general one that will make the positive examples derivable once the hypothesis is added to the background knowledge ($B \cup H \models E^+$) and is consistent with the negative examples ($\forall e^- \in E^- : B \cup H \not\models e^-$).

Mode declarations contains the schema for the allowed literals in the rule, and can be of the form $modeh(s)$ or $modeb(s)$ for the head or body of the rule respectively. The schema s is a grounded literal with placemarkers of the form '$+type$', '$-type$', or '$\#type$', with $type$ corresponding to the type of the literal's argument. The symbols '$+$', '$-$', and '$\#$' indicates whether the argument should be a variable in the head of the rule or one from previous body literals in the rule (input variable), a new fresh variable (output variable), or a constant respectively. Thus, the mode declarations $modeh(fly(+bird))$ and $modeb(wings(+bird, \#property, -int))$ would allow a rule such as $fly(X) \leftarrow wings(X, has\_flight\_feathers, Y)$ to be constructed, where X is a bird and Y is an integer, with $has\_flight\_feathers$ being a property of the bird's wings.

Predicate Invention is when the hypothesis includes a predicate that was neither within the background knowledge nor the examples. There are two reasons why a new predicate may be invented:

1. *Reformulation*: To identify interesting concepts not directly related to the learning goal that could be used to restructure the program. For example, if the background knowledge contains the rules:
   $pigeon(X) \leftarrow beak(X), feathers(X), wings(X), fly(X)$.
   $penguin(X) \leftarrow beak(X), feathers(X), wings(X), \neg fly(X)$.
   These rules share many conditions which could be factored out by inventing a new predicate $bird/1$, with the definition of $bird(X) \leftarrow beak(X), feathers(X), wings(X)$.

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 15–21

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The new predicate can then be used to replace all occurrences of the shared conditions within the background knowledge:
$pigeon(X) \leftarrow \underline{bird(X)}, fly(X).$
$penguin(X) \leftarrow \underline{bird(X)}, \neg fly(X).$
$\underline{bird(X)} \leftarrow beak(X), feathers(X), wings(X).$

2. *Bias Shift*: To specialise an overgeneral hypothesis and make it consistent with the examples. This is when the vocabulary available to the learner is not sufficient for constructing a consistent hypothesis. Therefore a new predicate is needed for specialising the overgeneral hypothesis such that it would no longer cover negative examples. Consider the following ILP problem:
$B = \{\ bird(alex).\ \ bird(bob).\ \}$ $E^+ = \{\ fly(alex).\ \}$ $E^- = \{\ fly(bob).\ \}$
$M = \{\ modeh(fly(+bird)).\ \}$
The current vocabulary is not strong enough to construct a consistent hypothesis. The mode declaration only allows the rule $fly(X)$, which is not sufficient for discriminating the negative example $fly(bob)$ from the positive one. Furthermore, no other conditions can be added to solve this problem given the current mode declaration. Thus, a new predicate p and the fact $p(alex)$ need to be added so that the consistent rule $fly(X) \leftarrow p(X)$ can be learned.

There are three main difficulties [13] that need to be considered when inventing new predicates:

1. *When to invent new predicate*
There needs to be a criteria for deciding when a new predicate is necessary, as we would not like to add useless predicates to the background knowledge that would only hinder the learner in future tasks.
2. *How to invent new predicate*
What structure should the new predicate have and how can its definition be found? Would a recursive call to the ILP algorithm in use be sufficient, or would a separate algorithm be required for learning the new predicate's definition?
3. *How to control the search space of the new predicate*
The search space for the new predicate could potentially be infinitely large, and the mode declarations for the original learning problem may no longer applies for learning the new predicate. There needs to be a way of limiting or directing the search space for the new predicate.

In this position paper, we discuss how a general framework for predicate invention, which is able to accommodate both cases of of theory reformulation and bias shift, could be developed. This takes advantage of recently proposed meta-level abductive approach to inductive logic programming, whereby inductive tasks are transformed into equivalent abductive task, reducing the computation to the search of equivalent abductive explanations. After a brief summary of the current state of art of the field, we show how a predicate invention task can be formulated as a meta-level search problem, that can compute new predicates for compacting the given background knowledge as well as specialising hypothesis.

2 Related Work

Past systems have concentrated on one of either reformulation or bias shift when inventing new predicates. DIALOGS [5], SIRIES [14] and CHAMP [12] all invents new predicate for bias shift, while INDEX [4] invents new predicates for reformulating the theory. Although in Cigol [11] the main goal is to find new rules for reformulation, its new predicates are only

invented when negative examples are confirmed by the oracle, the same situation as bias shift. There are also methods such as Statistical Predicate Invention (SPI) [9] and matrix sorting [8] that do not follow the framework of ILP, but are also able to introduce new predicates into existing theories. These methods find new predicates by identifying trends within the theory then grouping objects together, and the new predicates are introduced to restructure the theory according to those trends and groupings.

While each system's exact method is different from one another, they use the same strategy for inventing a new predicate. They start by identifying the appropriate structure of the new predicate then use it as an input to a learning algorithm, often the main inductive learning algorithm, to learn the definition of the new predicate. For instance, CHAMP invents its predicate by finding the smallest set of arguments that would completely discriminate the positive and negative examples, while Cigol finds the predicate's arguments by identifying non-unifiable arguments when generalising its examples. Having identified the structure, both systems then use it in a recursive call to their main learning algorithm.

The way each system controls the search space for the new predicate is more varied. Both DIALOGS and SIRIES prioritise some hypothesis structures over others, giving lower priorities to those that are more complicated or with new predicates. CHAMP and Cigol prefer a hypothesis that will achieve the most compression of their theory. While INDEX, SPI and matrix sorting uses some scoring mechanism for the most appropriate representation of their data.

3 Predicate invention at meta-level

Our framework is general enough for both bias shift and reformulation. Abstracting the problem to the meta-level would be suitable for inventing new predicates, as past methods have shown that meta-knowledge is often needed for deciding the new predicate's structure regardless of its purpose. ILP systems already have some means for reasoning about the language of its hypotheses by using the *top theory*, a theory on the encoding of the hypothesis as allowed by the mode declaration. Furthermore, the top theory should be flexible enough for importing any heuristics for inventing predicates.

Firstly, we briefly describe an existing ILP system called ASPAL that performs standard ILP tasks using meta-level abduction. We use this system to automate our framework as ASP uses sophisticated mechanisms for optimising search.

3.1 ASPAL

ASPAL (ASP Abductive Learning) is the Answer Set Programming (ASP), declarative programming based on stable model semantics [7], implementation of TAL. TAL (Top-directed Abductive Learning) [1, 2] is a top-down nonmonotonic ILP algorithm implemented in Prolog, using abductive learning to find the correct hypothesis. It solves an inductive problem by converting it into an abductive one. The hypothesis of the problem is found using a top theory, the theory concerning the construction of the hypothesis according to the mode declarations of the inductive problem. By reasoning with the top theory, TAL and ASPAL abstract the problem to the meta-level of the hypothesis' encoding.

$M = \{\ modeh(fly(+bird)).$
$\quad modeb(pigeon(+bird)).\}$

For example, using the mode declarations above, the corresponding top theory in ASPAL is as follows:

$$T_{ASPAL} = \{\ fly(X) \leftarrow bird(X), \$rule(r((fly, c, v))).$$
$$fly(X) \leftarrow bird(X), pigeon(X),$$
$$\$rule(r((fly, c, v), (pigeon, c, v(1)))).\ \}$$

The top theory matches the head of its clauses to examples of the ILP problem, testing conditions for those clauses and abducing the rule encoding $\$rule/1$ should no negative example satisfies the clause. Each tuple in $\$rule/1$ corresponds to a mode declaration (using fly or $pigeon$ for identification). The constants c and v are used to represents empty lists `[]` of constants and variables, while $v(1)$ represents the list `[1]` with a single index linking to the first variable in the rule. The constant list is used when the literal has constants in its arguments, while the variable list links variables in the rule together using their indexes. Using the top theory above with the background and examples:

$$B = \{\ bird(alex). \qquad E^+ = \{\ fly(alex).\ \}$$
$$bird(bob). \qquad E^- = \{\ fly(bob).\ \}$$
$$pigeon(alex).\ \}$$

The examples are used to construct an integrity constraint in the ASP program, such that all answer sets in the solution must include all positive examples and none of the negative ones. The first clause in T_{ASPAL} prevents the rule $fly(X)$ from being added to the hypothesis as the negative example $fly(bob)$ would also satisfy the clause. Thus $\$rule(r((fly, c, v)))$, the encoding for $fly(X)$, would not be included in the answer set. However, as the condition $pigeon(bob)$ is not satisfiable by the rule $fly(X) \leftarrow pigeon(X)$, its representation $\$rule(r((fly, c, v), (pigeon, c, v(1))))$ can be abduced.

ASPAL was used rather than TAL as the ASP solver is extremely efficient when solving a grounded program, ASPAL's current implementation avoids costly computation of the ASP grounder by using a preprocessor for constructing an ASP program with all possible grounded hypotheses. These advantages allow for many number of mode declarations to be used without high increase in computational time.

3.2 Predicate invention using meta-level abduction

In [10], a simple method for introducing new predicates through the mode declarations was shown by using *placeholders*. Placeholders are mode declarations of new predicates that were neither within the problem's example, its background knowledge, nor its mode declarations. For example, suppose we have the following problem:

$$B = \{\ alpha(a). \qquad E^+ = \{\ q(a, d), q(a, c).\ \}$$
$$alpha(b). \qquad E^- = \{\ q(c, d).\ \}$$
$$alpha(c). \qquad M = \{\ modeh(q(+alpha, +alpha)).\ \}$$
$$alpha(d).\ \}$$

To solve this using placeholders, we can add new mode declarations $modeh(new(\#alpha, \#alpha))$ and $modeb(new(+alpha, +alpha)$ to the problem, such that rules and facts such as $p(X, Y) \leftarrow new(X, X)$ and $new(a, a)$ can be learned. Running the problem in ASPAL takes only 0.01 seconds to solve. Should we want to add other seventeen alternative mode declarations for $modeb(new(+alpha, +alpha)$ (with negation and different combinations of constants, input and output variables), and limiting to maximum of one body literal per rule and two rules per hypothesis, ASPAL will solve the problem in 0.078 seconds and output 20 hypotheses. Many of the hypotheses subsumes each other, for instance:

$$H_1 = \{p(X, Y) \leftarrow new(X, Z). \quad new(a, a).\}$$
$$H_2 = \{p(X, Y) \leftarrow new(X, X). \quad new(a, a).\}$$
$$H_3 = \{p(X, Y) \leftarrow new(a, a). \quad new(a, a).\}$$

From the hypotheses above, simply selecting the shortest one is not sufficient as they all have the same length. Instead, we could lower the number of hypotheses by discarding

hypotheses that are subsumed by others. In the case above, H_2 and H_3 can be discarded as they are both subsumed by H_1, making H_1 the most general hypotheses out of the three hypotheses.

For reformulating a theory, [3] has shown how an ILP system can be used to revise theories by transforming the revisable section of the background knowledge and learning revision operators. Each revisable rule $r_i \leftarrow c_{i,1}, \ldots, c_{i,n}$ can be transformed to:

$r_i \leftarrow try(i, 1, c_{i,1}), \ldots, try(i, n, c_{i,n}), ext(i, r_i).$
$try(i, 1, c_{i,1}) \leftarrow c_{i,1}, use(i, 1).$
$try(i, 1, c_{i,2}) \leftarrow not\ use(i, 2).$
$\ldots$
$use(i, j) \leftarrow not\ del(i, j).$

Each $try/3$ clause is used to test if a condition j in a rule i is not used in the rule. Due to the definition of $use/2$, the condition $c_{i,j}$ is removed from the theory when the corresponding $del(i, j)$ is learned. The literal $ext/2$ is used for learning additional conditions to be added to the body in the rule. For instance, $ext(i, r_i) \leftarrow p(a)$ indicates that the rule r_i should be extended with the literal $p(a)$.

We have applied this method to reformulate the clauses:

$grandfather(X, Y) \leftarrow male(X), parent(Z, Y), parent(X, Z).$
$grandmother(X, Y) \leftarrow female(X), parent(Z, Y), parent(X, Z).$

As well as the mode declarations needed for revising the clauses, additional mode declarations were included such that the rule $new(X, Y) \leftarrow parent(Z, Y), parent(X, Z)$ can be learnt. This is so the learner can find a solution that would reformulate the rules to:

$grandfather(X, Y) \leftarrow male(X), \underline{new(X, Y)}.$
$grandmother(X, Y) \leftarrow female(X), \underline{new(X, Y)}.$
$\underline{new(X, Y)} \leftarrow parent(Z, Y), parent(X, Z).$

While we were able to acquire the above solution, the learner outputs many more solutions, with most not decreasing the size of the theory. This is because the search is only guided by the examples given, not by how much each hypothesis could compact the theory. Thus, to the learner, the following solution would be as good as the previous one:

$grandfather(X, Y) \leftarrow male(X), parent(Z, Y), \underline{new(X, Z)}.$
$grandmother(X, Y) \leftarrow female(X), parent(Z, Y), \underline{new(X, Z)}.$
$\underline{new(X, Y)} \leftarrow parent(X, Y).$

While comparing the literal count could help us identify the best revision, another simple solution is by using the optimisation feature of iClingo [6], the ASP solver used by ASPAL. As $del/2$ instances indicated removal of clauses, by asking the solver the find the maximum number of $del/2$ instances possible, we can then use it to find only solutions that will most reduce the size of the background knowledge. Similarly, finding the minimum number of new clauses added to the background knowledge can also help to find the solutions that will least increase the size of the background knowledge.

In conclusion, as well as the general ILP task, our framework is also capable performing the following tasks with predicate invention:

1. A bias shift task $\langle E, B, M \rangle$, where E is a set of examples, B is the background knowledge, and M is the set of mode declarations. A set of rules, a hypothesis H_B, is a solution to the task $\langle E, B, M \rangle$ if H_B is compatible with M extended by a new predicate p, $H_B \cup B$ is consistent with E, and H_B the predicate p such that $p \notin B$, $p \notin E$, and $p \notin M$.
2. A reformulation task $\langle B_N, B_R, M \rangle$, where B_N is the non-revisable background knowledge, B_R is the revisable background knowledge, and M is the set of mode declarations. A solution to the task $\langle B_N, B_R, M \rangle$ is tuple $H_R = \langle H_N, H_O \rangle$, where H_N (possibly empty)

is a set of new rules, and H_O is a sequence of revision operations, these include adding conditions to existing rules and deleting conditions or rules in B_R. H_R is a valid solution to $\langle B_N, B_R, M\rangle$ if H_N is compatible with M extended by a new predicate p, and for $o_1, \ldots, o_n \in H_O$: $B_R \otimes \{o_1, \ldots, o_n\} \cup B_N \cup H_N$ to have the same answer sets with $B_N \cup B_R$ for their shared predicates, and H_R may contain the predicate p such that $p \notin B_N \cup B_R$, and $p \notin M$.

3. By combining the previous two tasks, predicates can also be invented for correcting erroneous knowledge. Expanding the theory revision task to give a task $\langle E, B_N, B_R, M\rangle$, where E is a set of examples, B_N is the non-revisable background knowledge, B_R is the revisable background knowledge, M is the set of mode declarations, and $B_N \cup B_R$ is inconsistent with E. A solution to the task $\langle E, B_N, B_R, M\rangle$ is a tuple $H_T = \langle H_N, H_O\rangle$, where H_N is a set of new rules and H_O is a sequence of revision operations. H_T is a valid solution to $\langle E, B_N, B_R, M\rangle$ if H_N is compatible with M extended by a new predicate p, and for $o_1, \ldots, o_n \in H_O$: $B_R \otimes \{o_1, \ldots, o_n\} \cup B_N \cup H_N$ is consistent with E, and H_T contains a new predicate p such that $p \notin B_N \cup B_R$, $p \notin E$, and $p \notin M$.

4 Future Work

The objective of our research is a general ILP framework with capacity for an efficient way of inventing new predicates, able to invent new predicates for both reformulation and bias shift.

The simple approach of generating all placeholders seems to be able to handle all issues we outlined in Section 1: (i) placeholders can be added to the ASP program when the original learning problem fails to produce a hypothesis, (ii) use all possible placeholders, starting from one argument and increasing the number of its arguments until solutions are found, and rely on the learning algorithm of ASPAL to compute the hypotheses, (iii) search can be controlled by limiting the number of rules within the hypothesis and increasing it until the minimum number of rules is found. However, we still need to test it on larger problems to ensure that high number of mode declarations does not lead to a great increase in computational time or the number of solutions. If so, then we will need to find a way to limit the number of mode declarations that are considered for each time new predicates are needed.

For theory reformulation, we still need to determine when should the theory be reformulated, and how to control the search. A way to control the search is by associating each revision operator with a measure of its effect on the program size, so that the learner can use it to discard hypothesis that do not reduce the theory's size.

ASPAL already has some preliminary work done on hypothesis refinement, which could be used after the learner has gone through the search space from the given mode declarations. We plan to complete this hypothesis refinement framework of ASPAL, as this will help with determining when to invent new predicates for bias shift. It will allow us to invent new predicates only when demanded, similar to systems such as CHAMP and SIRIES.

Acknowledgements We would like to thank Domenico Corapi for his help and discussion on TAL and ASPAL.

References

1 Domenico Corapi. *Nonmonotonic Inductive Logic Programming as Abductive Search.* PhD thesis, Imperial College London, 2011.

2 Domenico Corapi, Alessandra Russo, and Emil Lupu. Inductive logic programming as abductive search. In Manuel Hermenegildo and Torsten Schaub, editors, *Technical Commu-*

nications of the 26th International Conference on Logic Programming, volume 2010, pages 54–63, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

3 Domenico Corapi, Alessandra Russo, Marina De Vos, Julian A. Padget, and Ken Satoh. Normative design using inductive learning. *TPLP*, 11(4-5):783–799, 2011.

4 Peter A Flach. Predicate Invention in Inductive Data Engineering. In P Brazdil, editor, *Machine Learning ECML93 European Conference on Machine Learning Proceedings*, volume 667 of *Lecture Notes in Artificial Intelligence*, pages 83–94. Springer-Verlag, 1993.

5 Pierre Flener. Inductive logic program synthesis with DIALOGS. In Stephen Muggleton, editor, *Inductive Logic Programming*, volume 1314 of *Lecture Notes in Computer Science*, pages 175–198. Springer Berlin / Heidelberg, 1997.

6 M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and S. Thiele. Engineering an incremental ASP solver. In M. Garcia de la Banda and E. Pontelli, editors, *Proceedings of the Twenty-fourth International Conference on Logic Programming (ICLP'08)*, volume 5366 of *Lecture Notes in Computer Science*, pages 190–205. Springer-Verlag, 2008.

7 Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. pages 1070–1080. MIT Press, 1988.

8 Charles Kemp, Joshua B Tenenbaum, Sourabh Niyogi, and Thomas L Griffiths. A probabilistic model of theory formation. *Cognition*, 114(2):165–196, 2010.

9 Stanley Kok and Pedro Domingos. Statistical predicate invention. In *Proceedings of the 24th International Conference on Machine Learning (2007)*, volume pages, pages 433–440. ACM Press, 2007.

10 Gregor Leban, Jure Zabkar, and Ivan Bratko. An Experiment in Robot Discovery with ILP. In *ILP 08 Proceedings of the 18th international conference on Inductive Logic Programming*, pages 77–90, 2008.

11 Stephen Muggleton and Wray L Buntine. Machine Invention of First Order Predicates by Inverting Resolution. In *Machine Learning*, pages 339–352, 1988.

12 B K M N M Shimura. Discrimination-Based Constructive Induction of Logic Programs. In *AAAI92 proceedings Tenth National Conference on Artificial Intelligence July 1216 1992*, page 44. Aaai Pr, 1992.

13 Irene Stahl. Predicate Invention in Inductive Logic Programming. In Luc De Raedt, editor, *Advances in Inductive Logic Programming*, pages 34–47. IOS Press, 1996.

14 Ruediger Wirth and Paul O'Rorke. Constraints on predicate invention. In Stephen H Muggleton, editor, *Proceedings of the Eighth International Workshop on Machine Learning*, pages 457–461. Morgan Kaufmann, 1991.

Targeting a Practical Approach for Robot Vision with Ensembles of Visual Features

Emanuela Boroş

Alexandru Ioan Cuza University
Faculty of Computer Science, Iaşi, Romania
emanuela.boros@info.uaic.ro

Abstract

We approach the task of topological localization in mobile robotics without using a temporal continuity of the sequences of images. The provided information about the environment is contained in images taken with a perspective colour camera mounted on a robot platform. The main contributions of this work are quantifiable examinations of a wide variety of different global and local invariant features, and different distance measures. We focus on finding the optimal set of features and a deepened analysis was carried out. The characteristics of different features were analysed using widely known dissimilarity measures and graphical views of the overall performances. The quality of the acquired configurations is also tested in the localization stage by means of location recognition in the Robot Vision task, by participating at the ImageCLEF International Evaluation Campaign. The long term goal of this project is to develop integrated, stand alone capabilities for real-time topological localization in varying illumination conditions and over longer routes.

1998 ACM Subject Classification I.2.10 Vision and Scene Understanding, I.4.3 Enhancement, I.4.6 Segmentation, I.4.10 Image Representation

Keywords and phrases Visual Place Classification, Robot Topological Localization, Visual Feature Detectors, Visual Feature Descriptors

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.22

1 Introduction and Related Work

Topological localization is a fundamental problem in mobile robotics. Most mobile robots must be able to locate itself in their environment in order to accomplish their tasks. Robot visual localization and place recognition are not easy tasks, and this is mainly due to the perceptive ambiguity of acquired data and the sensibility to noise and illumination variations of real world environments. In order to help reduce this gap, this work addresses the problem of topological localization of a robot that uses a single perspective camera in an office environment. The robot should be able to answer the question *where are you?* when presented with a test sequence representing a room category seen during training [25, 28, 17].

Many approaches during last years have been developed using different methods for robotic topological localization such as topological map building which makes good use of temporal continuity [30], simultaneous localization and mapping [8], using Monte-Carlo localization [32], appearance-based place recognition for topological localization, panoramic vision creation [31].

The problem of topological mobile localization has mainly three dimensions: a type of environment (indoor, outdoor, outdoor natural), a perception (sensing modality) and a localization model (probabilistic, basic). Numerous papers deal with indoor environments [30, 31, 10, 15] and a few deal with outdoor environments, natural or urban [29, 13]. Experimental

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 22–28

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

results for wide baseline image matching suggest the need for local invariant descriptors of images. Earlier research into invariant features focused on invariance to rotation and translation. There has also been research into the development of fully invariant features [5, 18, 19]. In his milestone paper [16], D. Lowe has proposed a scale-invariant feature transform (SIFT) for recognition based on local extrema of difference-of-Gaussian filters in scale-space that is invariant to image scaling and rotation, illumination and viewpoint changes. Lately, a new method has been proposed, Affine-SIFT (ASIFT) that simulates all image views obtainable by varying the two camera axis orientation parameters, namely the latitude and the longitude angles, left over by the SIFT method [21]. However, full affine invariance has not been achieved due partly to the impractically large computational cost. SIFT is a 128 dimensional feature vector that captures the spatial structure and the local orientation distribution of a region surrounding a keypoint. The SIFT method has been popularly applied for scene recognition [33, 1] and detection [11, 23] and robot localization [2, 24, 22].

We analyze the problem of topological localization without taking in consideration the use of the temporal continuity of the sequences of images which could be considered an advantage by adding an additional understanding of the space. Our approach represents an extension of our previous work [3, 4] where each RGB image is processed to extract sets of SIFT keypoints from where the descriptors are defined. In this paper the comparison is carried out for different configurations of features and matching distances of a topological localization system. We perform an exhaustive evaluation and introduce new analysis statistics between the quantization solutions.

2 Experimental Setup

2.1 Feature Matching

In this section we introduce different dissimilarity measures to compare features. That is, a measure of dissimilarity between two features and thus between the underlying images is calculated. Many of the features for images are in fact histograms (color histograms, invariant feature histograms, texture histograms, local feature histograms, and other feature histograms). As comparison of distributions is a well known problem, a lot of comparison measures have been proposed and compared before [26]. In the following, dissimilarity measures to compare two histograms H and K are proposed. Each of these histograms has n bins and H_i is the value of the i-th bin of histogram H.

- **Minkowski-form Distance** (L_1 distance is often used for computing dissimilarity between color images, also experimented in color histograms comparison [14]):

$$\mathrm{D}_{Lr}(H,K) = (\sum_{i=1} |H_i - K_i|)^{\frac{1}{r}} \tag{1}$$

- **Jensen Shannon Divergence** (also referred to as **Jeffrey Divergence** [9], is an empirical extension of the Kullback-Leibler Divergence. It is symmetric and numerically more stable):

$$\mathrm{D}_{JSD}(H,K) = \sum_{i=1} H_i \log \frac{2H_i}{H_i + K_i} + K_i \log \frac{2K_i}{K_i + H_i} \tag{2}$$

- χ^2 **Distance** (measures how unlikely it is that one distribution was drawn from the population represented by the other, [20]):

$$\mathrm{D}_{\chi^2}(H,K) = \sum_{i=1} \frac{(H_i - K_i)^2}{H_i} \tag{3}$$

- **Bhattacharyya Distance [7]** (measures the similarity of two discrete or continuous probability distributions). For discrete probability distributions H and K over the same domain, it is defined as:

$$\mathrm{D}_B(H,K) = -\ln \sum_{i=1} \sqrt{(H_i K_i)} \tag{4}$$

2.2 Datasets (Benchmark)

The chosen dataset contains images from nine sections of an office obtained from **CLEF (Conference on Multilingual and Multimodal Information Access Evaluation)**. Detailed information about the dataset are in the overview and ImageCLEF publications [25, 28, 17]. This dataset contains images that are widely used in topological localization image classification papers and it has already been split into three training sets of images, as shown in Table 1 one different from another. The provided images are in the RGB color space. The sequences are acquired within the same building and floor but there can be variations in the lighting conditions (sunny, cloudy, night) or the acquisition procedure (clockwise and counter clockwise).

Areas	Training1	Training2	Training3
Corridor	438	498	444
ElevatorArea	140	152	84
LoungeArea	421	452	376
PrinterRoom	119	80	65
ProfessorOffice	408	336	247
StudentOffice	664	599	388
TechnicalRoom	153	96	118
Toilet	198	240	131
VisioConference	126	79	60

Table 1 Training Sequences of An Office Environment.

2.3 Comparison of Different Distance Functions for Global Features

Global features capture the diagnostic structure of the image, an overall view of the image that is transformed in histograms of frequencies. Existing color-based general-purpose image retrieval systems as [27, 6] roughly fall into three categories depending on the signature extraction approach used: histogram, color layout, and region-based search. In this paper, histogram-based search methods are investigated in two different color spaces, RGB (**R**ed, **G**reen, and **B**lue) and HSV (**H**ue, **S**aturation, and **V**alue). RGB and HSV color histograms are subject to tests with Jeffrey Divergence, χ^2, Bhattacharyya, Minkowski and respectively

the widely used Euclidean distance measure. These were chosen considering the literature that underlies them as achieving the best results in image matching [7, 26].

The retrieved classes for images (*Corridor*, *LoungeArea* etc.) depend on a threshold, those below this value being rejected. This becomes an optimization problem of finding the best value that will cut the unwanted results, considering that it is better to have no results than inconsistent results. To accomplish this, we used the genetic algorithm explained in detail in [12]. For these experiments, we used a population of 200 individuals, the mutation probability of 0.15, and the crossover, of 0.7. The optimization process is stopped after 1000 generations. We used a selection scheme *rank selection* with *elitism*. For RGB histograms, as can be seen

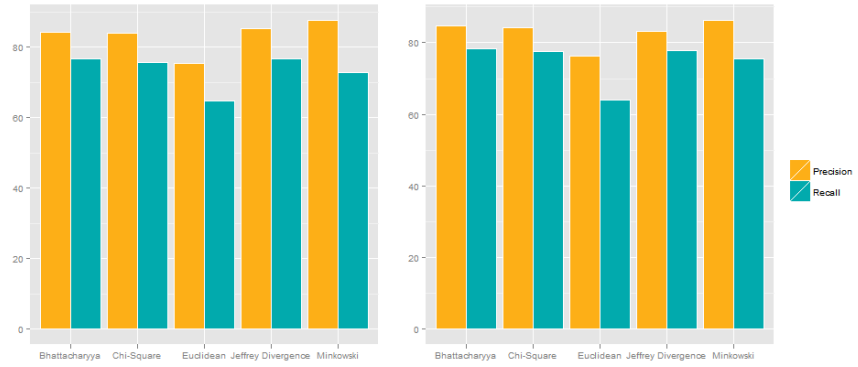


Figure 1 Precision and recall depending on measure distance (RGB & HSV Histograms).

in Figure 1, Bhattacharyya and Jeffrey Divergence obtained the highest recall and also, high precision, the highest F-measure being obtained by Jeffrey Divergence (0.806) extremely close to Bhattacharyya (0.802). The lowest performance is with Euclidean distance, having not only a low recall which means that this solution will bring more irrelevant results than using the other distances, but also a lower precision. In the case of using HSV histograms, the Bhattacharyya distance leaded to good results with a F-measure of 0.81 close to χ^2 distance with 0.807 and Minkowski with a F-measure of 0.805 . Folowing these chosen metrics, we adopted the vizualization with confusion matrices. Entries on the diagonal of the matrix, in blue, count the correct calls. Entries off the diagonal, in fading blue, count the misclassifications. Corresponding to the confusion matrix represented in Figure 2, the results show that HSV histogram with Bhattacharyya distance yielded very similar results with RGB choices of distances but clearly outperforms RGB histogram comparison with Jeffrey Divergence distance, similarity probabilty peaking at 100% in some of the office sections (*PrinterRoom, StudentOffice*).

2.4 Comparison of Different Distance Functions for Local Features

The two types of features used in the experiments are SIFT (Scale Invariant Feature Transform) and ASIFT (Affine Scale Invariant Feature Transform). The advantages of using these features are that they describe localized image regions (*patches*), the descriptors are computed around interest points, there is no need for segmentation and they are robust to occlusion and clutter. The disadvantage is that images are represented by different size sets of feature vectors and they do not lend themselves easily to standard classification techniques.

These results were obtained performing experiments on local feature histograms obtained using the *bag-of-visual-words* model. The descriptors are quantized and normalized. Different

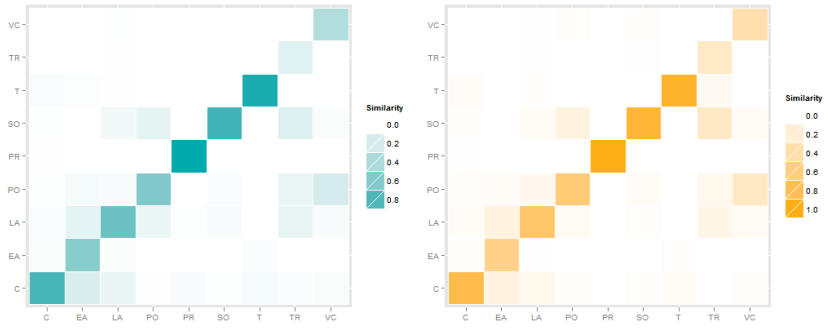


Figure 2 Confusion Matrix (RGB/HSV Histograms) using Jeffrey Divergence/Bhattacharyya Distances.

dissimilarity measures for the different types of features are compared experimentally and the performance for the different types of features is quantitatively measured.

For matthcing features, we chose literature-based distances known as having the best results: Euclidean, Minkowski, χ^2 and Jeffrey Divergence distances. For each of the local features descriptors we created Precision/Recall graphs from which we determine the superior runs. Figure 3 shows the Precision/Recall graphs for SIFT, respectively ASIFT and also shows that there is still vast room for improvement but the most promising results were obtained in the case of the usage of SIFT descriptors with Minkowski and Euclidean distance. The results show that Euclidean and Minkowski distance yielded very similar results, in the case of SIFT features matching.

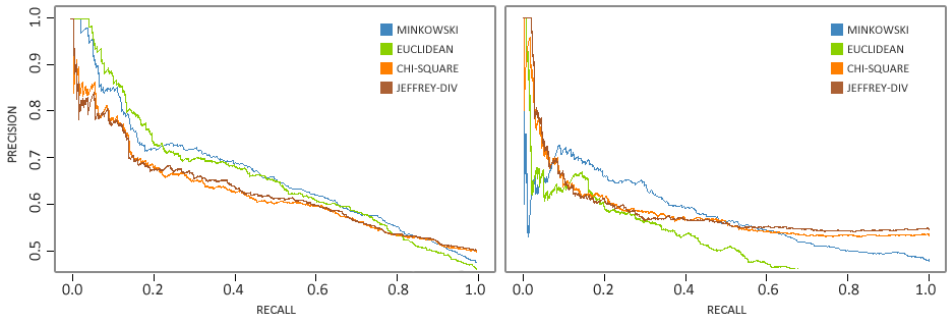


Figure 3 PR curves using different distance measures (SIFT & ASIFT).

3 Conclusions and Future Work

In this work, we approached the task of topological localization without using a temporal continuity of the sequences of images using a broad variety of features for image recognition. The provided information about the environment is contained in images taken with a perspective color camera mounted on a robot platform and it represents a know office environment dataset offered by ImageCLEF.

A large scale of global and local invariant features of images was presented, investigated, and experimentally evaluated. To analyze the features various dissimilarity measures were implemented and tested, as different features require different comparison methods.

The experiments show that the configurations from different feature descriptors and distance measures depends on the proper combinations. One important aspect is to use a selection of features accounting for the different properties of the images as there is no feature capable of covering all aspects of an image. The experiments showed the following features are suitable:

- RGB & HSV color histograms
- SIFT (Scale Invariant Feature Transform) as visual words with an Euclidean 100-means

The experiments showed also that the following image matching settings are suitable:

- RGB color histograms with Jeffrey Divergence distance & HSV color histograms with Bhattacharyya distance
- SIFT (Scale Invariant Feature Transform) matched with Minkowski distance

From the fact that most of the works cited are from the last couple of years, topological localization is a new and active area of research. which is increasingly producing interest and enforces further development. A first starting point for this field is given in this thesis, along with notable experimental results, but there is still room for improvement and further research.

References

1 M. Bennewitz, C. Stachniss, W. Burgard, and S. Behnke. Metric localization with scale-invariant visual features using a single perspective camera. *European Robotics Symposium 2006, ser. STAR Springer tracts in advanced robotics*, 22, 2006.

2 M. Bennewitz, C. Stachniss, W. Burgard, and S. Behnke. Metric localization with scale-invariant visual features using a single perspective camera. *European Robotics Symposium 2006*, 22 of STAR Springer tracts in advanced robotics:143–157, 2006.

3 E. Boroş, G. Roşca, and A. Iftene. Uaic: Participation in imageclef 2009 robotvision task. *Proceedings of the CLEF 2009 Workshop*, Sep 2009.

4 E. Boroş, G. Roşca, and A. Iftene. Using sift method for global topological localization for indoor environments. *Multilingual Information Access Evaluation II. Multimedia Experiments [Lecture Notes in Computer Science Volume 6242 Part II]*, 6242:277–282, 2009.

5 M. Brown and D.G Lowe. Invariant features from interest point groups. *The 13th British Machine Vision Conference, Cardiff University, UK*, pages 253–262, 2002.

6 R. Chakravarti. A study of color histogram based image retrieval. *Information Technology: New Generations, 2009. ITNG '09*, 2009.

7 E. Choi and C. Lee. Feature extraction based on the bhattacharyya distance. *Pattern Recognition*, 36:1703—1709, 2003.

8 H. Choset and K. Nagatani. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *IEEE Trans. Robot. Automat.*, 17(2):125–137, 2001.

9 T. Deselaers, D. Keysers, and H. Ney. Features for image retrieval: An experimental comparison. *Information Retrieval*, 2008.

10 G. Dudek and D. Jugessur. Robust place recognition using local appearance based methods. *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1030–1035, 2000.

11 G. Fritz, C. Seifert, M. Kumar, and L. Paletta. Building detection from mobile imagery using informative sift descriptors. *Lecture Notes in Computer Science*, pages 629–638, 2005.

12 A. L. Gînscă and A. Iftene. Using a genetic algorithm for optimizing the similarity aggregation step in the process of ontology alignment. *Proceedings, of 9th International Conference RoEduNet IEEE*, pages 118–122, Jun 2010.

13 J.-J. Gonzalez-Barbosa and S. Lacroix. Rover localization in natural environments by indexing panoramic images. *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1365–1370, 2002.

14 A. B. Kurhe, S. S. Satonka, and P. B. Khanale. Color matching of images by using minkowski- form distance. *Global Journal of Computer Science and Technology, Global Journals Inc. (USA)*, 11, 2011.

15 L. Ledwich and S. Williams. Reduced sift features for image retrieval and indoor localisation. *Australasian Conf. on Robotics and Automation*, 2004.

16 D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2(60):91–110, 2004.

17 W. Lucetti and E. Luchetti. Combination of classifiers for indoor room recognition, cgs participation at imageclef2010 robot vision task. *Conference on Multilingual and Multimodal Information Access Evaluation (CLEF 2010)*, 2010.

18 K. Mikolajczyk and C. Schmid. An afine invariant interest point detector. *Proceedings of the 7th European Conference on Computer Vision*, pages 128–142, 2002.

19 K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *IJCV*, 60(1), 2004.

20 K. Mikolajczyk, C. Schmid, H. Harzallah, and J. van de Weijer. Learning object representations for visual object class recognition. *Visual Recognition Challange*, 2007.

21 J. Morel and G. Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009.

22 A. Murarka, J. Modayil, and B. Kuipers. Building local safety maps for a wheelchair robot using vision and lasers. *Proceedings of the The 3rd Canadian Conference on Computer and Robot Vision*, 2006.

23 A. Negre, H. Tran, N. Gourier, D. Hall, A. Lux, and JL Crowley. Comparative study of people detection in surveillance scenes. structural, syntactic and statistical pattern recognition. *Proceedings Lecture Notes in Computer Science*, 4109:100–108, 2006.

24 D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. *CVPR*, 2:2161–2168, 2006.

25 A. Pronobis, O. M. Mozos, B. Caputo, and P. Jensfelt. Multi-modal semantic place classification. *Int. J. Robot. Res.*, 29(2-3):298–320, February 2010.

26 J. Puzicha, Y. Rubner, C. Tomasi, and J. Buhmann. Empirical evaluation of dissimilarity measures for color and texture. *Proc. International Conference on Computer Vision, Vol. 2*, pages 1165–1173, 1999.

27 J. Sangoh. Histogram-based color image retrieval. *Psych221/EE362 Project Report*, 2001.

28 O. Saurer, F. Fraundorfer, and M. Pollefeys. Visual localization using global visual features and vanishing points. *Conference on Multilingual and Multimodal Information Access Evaluation (CLEF 2010)*, 2010.

29 Y. Takeuchi and M. Hebert. Finding images of landmarks in video sequences. *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 1998.

30 S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99:21–71, February 1998.

31 I. Ulrich and I. Nourbakhsh. Appearance-based obstacle detection with monocular color vision. *Proceedings of AAAI Conference*, pages 866–871, 2000.

32 J. Wolf, W. Burgard, and H. Burkhardt. Robust vision-based localization for mobile robots using an image retrieval system based on invariant features. *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

33 J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV*, 73(2):213–238, Jun 2007.

Incremental HMM with an improved Baum-Welch Algorithm

Tiberiu S. Chis[1] and Peter G. Harrison[2]

1,2 Department of Computing, Imperial College London
South Kensington Campus, London, UK
tiberiu.chis07@imperial.ac.uk , pgh@doc.ic.ac.uk

Abstract

There is an increasing demand for systems which handle higher density, additional loads as seen in storage workload modelling, where workloads can be characterized on-line. This paper aims to find a workload model which processes incoming data and then updates its parameters "on-the-fly." Essentially, this will be an incremental hidden Markov model (IncHMM) with an improved Baum-Welch algorithm. Thus, the benefit will be obtaining a parsimonious model which updates its encoded information whenever more real time workload data becomes available. To achieve this model, two new approximations of the Baum-Welch algorithm are defined, followed by training our model using discrete time series. This time series is transformed from a large network trace made up of I/O commands, into a partitioned binned trace, and then filtered through a K-means clustering algorithm to obtain an observation trace. The IncHMM, together with the observation trace, produces the required parameters to form a discrete Markov arrival process (MAP). Finally, we generate our own data trace (using the IncHMM parameters and a random distribution) and statistically compare it to the raw I/O trace, thus validating our model.

1998 ACM Subject Classification G.3 Probability and Statistics

Keywords and phrases hidden Markov model, Baum-Welch algorithm, Backward algorithm, discrete Markov arrival process, incremental workload model

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.29

1 Introduction

A hidden Markov model (HMM) is a bivariate Markov chain which encodes information about the evolution of a time series. First developed by Baum and Petrie in 1966 [1], HMMs can faithfully represent workloads for discrete time processes and therefore be used as portable benchmarks to explain and predict the complex behaviour of these processes. When constructing a HMM, the three main problems that need to be addressed are: First, given the model parameters, compute the probability that the HMM generates a particular sequence of observations, solved by the *Forward-Backward algorithm*; Second, given a sequence of observations, find the most likely set of model parameters, solved by statistical inference through the *Baum-Welch algorithm*, which uses the Forward-Backward algorithm; Third, find the path of hidden states that is most likely to generate a sequence of observations, solved using a posteriori statistical inference in the *Viterbi algorithm*. In this paper, we propose an incremental variation of the Baum-Welch algorithm by creating two approximations of the Forward-Backward algorithm. This way, we will be able to process incoming I/O trace data incrementally and update our HMM parameters "on-the-fly" as new trace data becomes available. The HMM which uses this incremental Baum-Welch algorithm (IncHMM) produces the required parameters to form a discrete Markov arrival process (MAP), which we

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 29–34

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

refer to as our Workload Model. For our results, we validate two Workload Models using averages from the raw and IncHMM-generated traces. Finally, we compare our results with current work in the field, identifying any improvements for the future.

2 Forward-Backward Algorithm

The Forward-Backward algorithm, which is used in our incremental Baum-Welch algorithm, solves the following problem: Given the observations $O = (O_1, O_2, \ldots, O_T)$ and the model $\lambda = (A, B, \pi)$[1], calculate $P(O \mid \lambda)$ (i.e. the probability of the observation sequence given the model), and thus determine the likelihood of O. Based on the solution in [5], we explain the "Forward" part of the algorithm, which is the α-pass, followed by the "Backward" part or the β-pass. We define the forward variable $\alpha_t(i)$ as the probability of the observation sequence up to time t and of state q_i at time t, given our model λ. In other words, $\alpha_t(i) = P(O_1, O_2, \ldots, O_t, s_t = q_i \mid \lambda)$, where $i = 1, 2, \ldots, N$, N is the number of states, $t = 1, 2, \ldots, T$, T is the number of observations, and s_t is the state at time t. The solution of $\alpha_t(i)$ is inductive:

1. Initally, for $i = 1, 2, \ldots, N$: $\alpha_1(i) = \pi_i b_i(O_1)$
2. Then, for $i = 1, 2, \ldots, N$ and $t = 2, 3, \ldots, T$: $\alpha_t(i) = [\sum_{j=1}^{N} \alpha_{t-1}(j) a_{ji}] b_i(O_t)$
   where $\alpha_{t-1}(j) a_{ji}$ is the probability of the joint event that $O_1, O_2, \ldots O_{t-1}$ are observed (given by $\alpha_{t-1}(j)$) and there is a transition from state q_j at time $t-1$ to state q_i at time t (given by a_{ji}), and also $b_i(O_t)$ is the probability that O_t is observed from state q_i.
3. It follows that: $P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$
   where we used the fact that $\alpha_T(i) = P(O_1, O_2, \ldots, O_T, s_T = q_i \mid \lambda)$.

Similarly, we can define the backward variable, $\beta_t(i)$ as the probability of the observation sequence from time $t+1$ to the end, given state q_i at time t and the model λ. Then, $\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots O_T \mid s_t = q_i, \lambda)$ and the recursive solution is:

1. Initially, for $i = 1, 2, \ldots, N$: $\beta_T(i) = 1$
2. Then, for $i = 1, 2, \ldots, N$ and $t = T-1, T-2, \ldots, 1$: $\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$
   where we note that the observation O_{t+1} can be generated from any state q_j.

With the α and β values now computed, we attempt to create an incremental version of the Baum-Welch algorithm, which will use both of these values.

3 Incremental Baum-Welch Algorithm

Given the model $\lambda = (A, B, \pi)$, the Baum-Welch algorithm (BWA) trains a HMM on a fixed set of observations $O = (O_1, O_2, \ldots, O_T)$. By adjusting its parameters A, B, π, the BWA aims to maximise $P(O \mid \lambda)$. As explained in Section 2.3.2 of [6], the parameters of the BWA are updated iteratively by the following formulas:

1. Initially, for $i = 1, 2, \ldots, N$: $\pi'_i = \gamma_1(i)$
2. For A: $a'_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{j=1}^{N} \sum_{t=1}^{T-1} \xi_t(i,j)}$

[1] A is the state transition matrix, B is the observation matrix, and π is the initial state distribution.

3. For B: $b_j(k)' = \frac{\sum_{t=1, O_t=k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$

   where $\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)}$ and $\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$

We can now re-estimate our model using $\lambda' = (A', B', \pi')$ where $A' = \{a'_{ij}\}$, $B' = \{b_j(k)'\}$ and $\pi' = \{\pi'_i\}$. However, this re-estimation only works on a fixed set of observations, and a useful upgrade for the BWA would be to handle infrequent, higher density, additional loads mainly for on-line characterization of workloads [2]. To have an incremental HMM automatically updating its parameters as more real time workload data becomes available would achieve this, as well as consistently analyse processes over time in a computationally efficient manner. This new model will be a hybrid between a standard HMM and an incremental HMM which updates the current parameters A, B, π based on the new set of observations. Therefore, after the standard HMM has finished training on its observation set, we aim to calculate the α, β, ξ and γ variables on the new incoming set of observations. For example, if we have trained a HMM on T observations and wish to add new observations to update our model incrementally, we notice that $\alpha_{T+1}(i) = [\sum_{j=1}^{N} \alpha_T(j) a_{ji}] b_i(O_T)$. Since we possess the values of $\alpha_T(j)$, a_{ji} and $b_i(O_T)$, the new α values can be computed quite easily using the forward recurrence formula. However, to find $\beta_{T+1}(i)$ is not so easy as it relies on the backward formula with a one step lookahead $\beta_{T+1}(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{T+2}) \beta_{T+2}(j)$ and unfortunately we do not have $\beta_{T+2}(j)$. Therefore an approximation for the β variables is needed, preferrably a forward recurrence formula similar to the α formula. The new ξ and γ variables (and therefore the new entries a'_{ij} and $b_j(k)'$) can be calculated easily once we have the complete α and β sets. Building on previous work seen in Section 4.8.3 of [6], we attempt to find several new approximations for the β values.

3.1 First β Approximation

The first approximation for the β variables will assume that, at time t and for state i, we have that $\beta_t(i) = \delta(t,i)$ is a decay function which tends to 0 as $t \to 0$. Therefore, for a sufficiently large observation set and at a sufficiently small t, we obtain the approximate result $\delta(t,i) - \delta(t,j) \approx 0$, where i and j are different states. This then gives the near equality $\delta(t,i) \approx \delta(t,j)$ and hence by our earlier assumption we have the important approximation:

$$\beta_t(i) \approx \beta_t(j) \tag{1}$$

Let us now transform our β recurrence formula $\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$ into matrix form, using the notation $b_j = b_j(O_{t+1})$ for ease of use. Since we are using two states in our Workload Model, we set $N = 2$. It then follows that

$$\begin{pmatrix} \beta_t(1) \\ \beta_t(2) \end{pmatrix} = \begin{pmatrix} a_{11}b_1 & a_{12}b_2 \\ a_{21}b_1 & a_{22}b_2 \end{pmatrix} \begin{pmatrix} \beta_{t+1}(1) \\ \beta_{t+1}(2) \end{pmatrix}$$

then pre-multiply by $\begin{pmatrix} \alpha_t(1) & \alpha_t(2) \end{pmatrix}$:

$$\begin{pmatrix} \alpha_t(1) & \alpha_t(2) \end{pmatrix} \begin{pmatrix} \beta_t(1) \\ \beta_t(2) \end{pmatrix} = \begin{pmatrix} \alpha_t(1) & \alpha_t(2) \end{pmatrix} \begin{pmatrix} a_{11}b_1 & a_{12}b_2 \\ a_{21}b_1 & a_{22}b_2 \end{pmatrix} \begin{pmatrix} \beta_{t+1}(1) \\ \beta_{t+1}(2) \end{pmatrix}$$

and multplying out we get

$$\sum_{i=1}^{2} \alpha_t(i) \beta_t(i) = \begin{pmatrix} \alpha_t(1)a_{11}b_1 + \alpha_t(2)a_{21}b_1 & \alpha_t(1)a_{12}b_2 + \alpha_t(2)a_{22}b_2 \end{pmatrix} \begin{pmatrix} \beta_{t+1}(1) \\ \beta_{t+1}(2) \end{pmatrix}$$

where by defintion of $\alpha_{t+1}(i)$ it follows that

$$\sum_{i=1}^{2} \alpha_t(i)\beta_t(i) = \begin{pmatrix}\alpha_{t+1}(1) & \alpha_{t+1}(2)\end{pmatrix} \begin{pmatrix}\beta_{t+1}(1) \\ \beta_{t+1}(2)\end{pmatrix}$$

We notice that $\sum_{i=1}^{2} \alpha_t(i)\beta_t(i) = P(O \mid \lambda) = \sum_{i=1}^{2} \alpha_T(i)$ where T is the total number of observations. Quite fittingly, the term $P(O \mid \lambda)$ is already calculated for us from the α-pass. Finally, assuming that $t+1$ is sufficiently small and using (1) we can deduce that $\beta_{t+1}(1) \approx \beta_{t+1}(2)$, giving us

$$P(O \mid \lambda) \approx \begin{pmatrix}\alpha_{t+1}(1) & \alpha_{t+1}(2)\end{pmatrix} \begin{pmatrix}\beta_{t+1}(1) \\ \beta_{t+1}(1)\end{pmatrix}$$

we then factor out $\beta_{t+1}(1)$

$$P(O \mid \lambda) \approx \beta_{t+1}(1) \begin{pmatrix}\alpha_{t+1}(1) & \alpha_{t+1}(2)\end{pmatrix} \begin{pmatrix}1 \\ 1\end{pmatrix}$$

and multiply out the RHS

$$P(O \mid \lambda) \approx \beta_{t+1}(1)[\alpha_{t+1}(1) + \alpha_{t+1}(2)]$$

which gives our final approximation result:

$$\beta_{t+1}(1) \approx \beta_{t+1}(2) \approx \frac{P(O \mid \lambda)}{\sum_{i=1}^{2} \alpha_{t+1}(i)} \tag{2}$$

The β approximation seen in (2) produced very good results in our simulation. To achieve this simulation, we obtained a network trace (aka raw trace) from NetApp servers made up of timestamped I/O commands (single Common Internet File System *reads* and *writes*). We then partitioned this raw trace into one second intervals (aka binned trace) counting the number of reads and writes present in each interval or "bin". This binned trace was then filtered through a K-means clustering algorithm (assigning 7 clusters, i.e. K=7) and we obtained a discrete time series (aka observation trace) where each point is an integer between 1 and 7. This observation trace was given as a training set of 7000 points (i.e. 7000 seconds) to a HMM. Afterwards, 3000 new observations were added to this set, evaluating the 3000 points using our new β approximation. Thus, we were able to create the IncHMM, which stored information on 10000 consecutive observation points. Statistics on a raw trace of 10000 observations were compared with those of an IncHMM-generated trace (using our model parameters A, B, π and a random distribution to generate this trace) also of size 10000. The results are summarised below in Figure 1:

Reads/bin	Writes/bin
Raw Mean: 111.350	Raw Mean: 0.382
IncHMM Mean: 111.278	IncHMM Mean: 0.366
Raw Std Dev: 254.942	Raw Std Dev: 0.550
IncHMM Std Dev: 255.039	IncHMM Std Dev: 0.461

Figure 1 Statistics for raw and IncHMM traces using the first β approximation.

Figure 1 is divided into Reads/bin and Writes/bin to simplify analysis, where the bin is simply a one second interval. For example, a "Raw Mean of 111.350 Reads/bin" means that the raw I/O trace produced on average 111.350 read commands per second. Similarly, we

analyse the average number of writes per second as our I/O trace contains both reads and writes. Therefore, we can see from Figure 1 that the statistics for raw reads and IncHMM reads match extremely well, almost identical over the 10000 points. For the writes, there is a higher difference in the standard deviations than in the means. This is possibly due to a significant drop in the number of write procedures presented by the I/O trace, which the IncHMM did not reproduce entirely when generating its trace.

3.2 Second β Approximation

As before, we begin with the following vectors and the 2×2 transformation matrix (D):

$$\begin{pmatrix} \beta_t(1) \\ \beta_t(2) \end{pmatrix} = \begin{pmatrix} a_{11}b_1 & a_{12}b_2 \\ a_{21}b_1 & a_{22}b_2 \end{pmatrix} \begin{pmatrix} \beta_{t+1}(1) \\ \beta_{t+1}(2) \end{pmatrix}$$

where we use $b_i = b_i(O_{t+1})$, for ease of notation.

We then pre-multiply by the inverse of the transformation matrix (D^{-1}):

$$\begin{pmatrix} a_{11}b_1 & a_{12}b_2 \\ a_{21}b_1 & a_{22}b_2 \end{pmatrix}^{-1} \begin{pmatrix} \beta_t(1) \\ \beta_t(2) \end{pmatrix} = I_2 \begin{pmatrix} \beta_{t+1}(1) \\ \beta_{t+1}(2) \end{pmatrix}$$

where $D^{-1}D = I_2$ and I_2 is the 2×2 identity matrix.

By using Gauss-Jordan elimination to work out D^{-1}, the final equation is

$$\begin{pmatrix} \beta_{t+1}(1) \\ \beta_{t+1}(2) \end{pmatrix} = \frac{1}{b_1 b_2 (a_{11}a_{22} - a_{21}a_{12})} \begin{pmatrix} a_{22}b_2 & -a_{12}b_2 \\ -a_{21}b_1 & a_{11}b_1 \end{pmatrix} \begin{pmatrix} \beta_t(1) \\ \beta_t(2) \end{pmatrix}$$

where $b_1 \neq 0$, $b_2 \neq 0$ and $a_{11}a_{22} \neq a_{21}a_{12}$.

In the case that $b_i = 0$ for a state i, D has a column of all zero values, which means that D^{-1} cannot exist, and therefore a simple approximation for $\beta_{t+1}(i)$ is needed here. Considering all three cases, we present the full set of equations in (3). Underneath this, Figure 2 summarises the results of the simulation with the β approximation from (3):

$$\begin{pmatrix} \beta_{t+1}(1) \\ \beta_{t+1}(2) \end{pmatrix} = \begin{cases} \begin{pmatrix} 1.0 \\ \frac{\beta_t(2)}{a_{22}b_2} \end{pmatrix}, & \text{if } b_1 = 0 \\ \begin{pmatrix} \frac{\beta_t(1)}{a_{11}b_1} \\ 1.0 \end{pmatrix}, & \text{if } b_2 = 0 \\ D^{-1} \begin{pmatrix} \beta_t(1) \\ \beta_t(2) \end{pmatrix}, & \text{if } b_1 \neq 0, b_2 \neq 0, a_{11}a_{22} \neq a_{21}a_{12} \end{cases} \tag{3}$$

Reads/bin	Writes/bin
Raw Mean: 111.350	Raw Mean: 0.382
IncHMM Mean: 110.231	IncHMM Mean: 0.357
Raw Std Dev: 254.942	Raw Std Dev: 0.550
IncHMM Std Dev: 254.155	IncHMM Std Dev: 0.463

Figure 2 Statistics for raw and IncHMM traces using the second β approximation.

The results obtained were satisfying, including the reads which performed very well. In comparison, the writes slightly underperformed, possibly due to the read-dominated trace or perhaps a slight misjudgement by our clustering algorithm.

4 Conclusion and Future Work

The β approximations used in this paper have been successful after statistical comparisons between raw and IncHMM-generated traces. Thus, we have created two Workload Models (each with their own β approximation) which characterize data traces incrementally. Analysing current work in this field, for example Stenger et al. in 2001 [4] (where all new β variables were given a value of 1), it is clear that either Workload Model provides a better β approximation. When comparing our models with the incremental HMM from [3], all three models produced accurate results, except the latter had a backward formula that was not recursive in terms of the β values. A general improvement to our models would be to increase the accuracy for the standard deviation of the IncHMM writes. This may be achieved by using significantly more observations from our I/O trace to obtain a greater variation in write entries. Perhaps adjusting the K parameter for our K-means clustering algorithm might also improve our results. Finally, we could test the IncHMM with another discrete time data trace, for example using a binned trace of hospital arrival times which stores the number of patients arriving every hour. Then, by choosing the most accurate β approximation of the two, we would obtain an incremental Workload Model capable of analysing a variety of discrete time series.

References

1 Baum, L. E., Petrie, T, *Stastical Inference for Probabilistic Functions of Finite Markov Chains*. In *The Annals of Mathematical Statistics*, **37**, pp. 1554-63, 1966

2 Harrison, P. G., Harrison, S. K., Patel N. M., Zertal, S. *Storage Workload Modelling by Hidden Markov Models: Application to Flash Memory*, In: *Performance Evaluation*, **69**, pp. 17-40, 2012

3 Florez-Larrahondo, G., Bridges, S., Hansen, E. A., *Incremental Estimation of Discrete Hidden Markov Models on a New Backward Procedure*, Department of Computer Science and Engineering, Mississippi State University, Mississippi, USA, 2005

4 Stenger, B., Ramesh, V., Paragois, N., Coetzee, F., Buhmann, J. M., *Topology free Hidden Markov Models: Application to background modeling*, pp. 297-301, Proceedings of the International Conference on Computer Vision, 2001

5 Rabiner, L. R., Juang, B. H., *An Introduction to Hidden Markov Models*. In *IEEE ASSP Magazine*, **3**, pp. 4-16, January, 1986

6 Chis, Tib, *Hidden Markov Models: Applications to Flash Memory Data and Hospital Arrival Times*, Department of Computing, Imperial College London, London, UK, 2011

Device specialization in heterogeneous multi-GPU environments

Gabriele Cocco[1] **and Antonio Cisternino**[1]

1 **Computer Science Dept., University of Pisa**
Largo Bruno Pontecorvo, Pisa, Italy
cocco@di.unipi.it, cisterni@di.unipi.it

Abstract

In the last few years there have been many activities towards coupling CPUs and GPUs in order to get the most from CPU-GPU heterogeneous systems. One of the main problems that prevent these systems to be exploited in a device-aware manner is the CPU-GPU communication bottleneck, which often doesn't allow to produce code more efficient than the GPU-only and the CPU-only counterparts. As a consequence, most of the heterogeneous scheduling systems treat CPUs and GPUs as homogeneous nodes, electing map-like data partitioning to employ both these processing resources. We propose to study how the radical change in the connection between GPU, CPU and memory characterizing the APUs (Accelerated Processing Units) affect the architecture of a compiler and if it is possible to use all these computing resources in a device-aware manner. We investigate on a methodology to analyze the devices that populate heterogeneous multi-GPU systems and to classify general purpose algorithms in order to perform near-optimal control flow and data partitioning.

1998 ACM Subject Classification C.1.3 Other Architecture Styles, D.1.3 Concurrent Programming, D.2.8 Metrics

Keywords and phrases HPC, APU, GPU, GPGPU, Heterogeneous computing, Parallel computing, Task scheduling

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.35

1 Introduction

In the last few years various researches demonstrated that GPU computing power isn't suitable to accelerate many algorithms[4], mostly due to the execution model and to the limited performances of the interconnection between the GPUs and the rest of the system. From the execution model point of view, SIMD doesn't fit very well with some computations, such as algorithms containing many branches and fine-grained data-parallel computations. Among the algorithms falling into these categories we can find Huffman Coding[6, 7] and KD trees construction[8]. To get the highest performances, the characteristics of the CPUs, such as wide caches and Multiple Thread Multiple Data (MTMD) execution model, should be employed to accelerate portions of such kind of algorithms. Unfortunately, partitioning algorithms to run heterogeneously on CPU-GPU systems is hold by the CPU-GPU interconnection and communication performance[2]. Whereas an algorithm may benefit to be carefully staged across the two computing resources, the time spent in transferring data often outweighs the time saved in executing code on the most specific resource. Given this, recent researches have mostly focused on exploiting the CPU and the GPU in an homogeneous way, ignoring the specific characteristics of each computing resource and partitioning data in a task-farm manner instead of scheduling differents parts of the control flow. S. Venkatasubramanian

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 35–41

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

and R. W. Vuduc [9] propose a solution to employ heterogeneous CPU-GPU platforms to accelerate Jacobi's iterative method for 2D Poisson equations. Since the target of this work is quite specific, hand-crafted implementations for CPU and GPU are proposed instead of a methodology to partition more general stencil algorithms and to automatically produce target code. C. Luk and S.H.H. Kim [11] present an entire programming system for CPU-GPU platforms where mapping between tasks and processing resources can be either performed by the programmer or automatically by the scheduler. While manual mapping may allow to partition control flow in addition to data, it substatially charges the programmer of determining a good partitioning strategy and hand-coding the implementations for both CPU and GPU. In [10], a machine learning approach is employed to statically decide a near-optimal scheduling strategy. Like in the other works, partitioning is based only on data and predictors are used to determine the amount of data to schedule to each processing element instead of which part of the algorithm to execute on it. In [14] a performance-history based scheduler is proposed to schedule tasks on the computing resource that demonstrated to be the most efficient in executing those tasks during previous executions. Since tasks are not analyzed nor classified on the bases of their features but are instead considered as blackboxes of which only the completion time is known, the scheduler requires to be retrained every time it is ported to a different CPU-GPU platform. E. Hermann et al. present a task scheduling approach for interactive physics simulations that allows to split tasks across multiple CPUs and GPUs[12] on the basis of task size and estimated completion time.

With the introduction of APUs, such as the Intel Ivy Bridge® and the AMD Fusion® family, the CPU-GPU communication performance has decisively increased[1], thanks to a novel architectural interconnection that overcomes the limits of the PCI-express bus and to the chance for the CPU and the GPU to effectively share data without the need for copies. APUs may therefore raise the chance for algorithms to be partitioned across heterogeneous processing resources in a device-specific manner. At the same time, given the different balances between computing and communication performance of integrated versus discrete GPUs, APU's and discrete GPUs in a hybrid multi-GPU system can be indepedently specialized to accelerate different kind of computations.

Our work consists in leveraging on the APU's capabilities to investigate on the chances to specialize both the on-chip resources and discrete GPUs in order to schedule portions of data and control flow on the bases of the specific characteristics of each device. Since APUs represent a relatively recent architecture, there are currently few resources on them, such as costs models, performance analysis or researches towards scheduling strategies to exploit this kind of tightly coupled platforms. S. Keely[8] discusses about adaptive mapping kd-tree construction on APUs to get the most from both the CPU and the integrated GPU, showing that on-chip communication performance allows to reduce the penalities of synchronization and host-device data transfer costs. K. Spafford et al.[5] propose an extensive comparison of various algorithms running on discrete versus APU's GPUs, illustrating the benefits of tighter coupling when data exchange becomes a dominant portion of the runtime. M. Daga at al. show a similar comparison[1], but int this case the intergrated GPU and a discrete one are tested using two different platforms. Since the two GPUs do not share the CPU executing the host code, comparing their performances is difficult and not fully reliable.

2 Methodology

The main aspects that characterize APUs is an high CPU-GPU communication performance and a computing power usually much lower than the computing power of mainstream discrete

GPUs. Given this, computations for which the time spent in CPU-GPU communication outclasses computing time should benefit of integrated GPU execution. The first step of our work consists in verifying this idea using a set of algorithms with different data-transfer/operations ratio (section 3.1).

The second step of the research consists in defining a reliable metric for classification of algorithms in terms of the device (integrated or discrete GPU) that is more suitable for their execution. We plan to train this metric on a particular system using a set of samples (i.e. popular and widely used parallel programming patterns) and taking into account the characteristics of the various devices in order to obtain a convenience threshold. Algorithms for which the value of the metric is below the threshold should be scheduled on the APU's GPU while algorithms for which the metric is above the threshold should be run on the discrete GPU. Following the results of the first step, the classification metric should be mainly based on the amount of data transferred and on the amount of operations executed on data. The term "operations" may refer to device ISA or intermediate language (PTX, AMD IL) instructions, to higher-level assembly instructions (MSIL, LLVM), to source-language-based operations (C/C++ assignments, arithmetic operations, etc.) or to language-unaware algorithmic complexity. Even if moving from device ISA to virtual machine assembly affects the amount of instructions executed, the number of operations considered in the metric should not necessarily match instructions executed by a device but instead characterize the complexity of the algorithm in a device-unaware manner. Moreover, as shown in section 3, the relative integrated versus discrete GPU performance seems to be correlated to the complexity of the (sequential) algorithm, which suggests the reliability of expressing operations in terms of sequential algorithms. The first definition of this metric, based on language-unaware algorithmic complexity, is discussed in section (3.2). For future work we plan to move to LLVM and to analyze LLVM code to determine the value of the metric for generic algorithms. LLVM provides various tools and services, such as loop informations, dead-code elimination and more, that can be exploited to simplify code analysis.

In addition to the operations executed and to the amount of data transferred other aspects may affect the GPUs relative performance, such as synchronization, branching, number of working threads and memory conflicts. While most of these aspects might be taken into account, we decide to start with a very simple metric and to refine it introducing additional parameters as soon as we encounter examples for which the metric fails in modeling the integrated versus discrete GPU performance. In addition, some of these features, like synchronizations and memory conflicts, are not exposed at LLVM level. Whenever the metric requires to take them into account, an OpenCL implementation of the algorithm should be available or it should be generated starting from LLVM implementation. AMD is currently developing an LLVM backend to produce AMD IL binaries, while an LLVM to Nvidia PTX is already available [19].

Founding our work on a device-unaware analysis of operations allows to overcome the dependency of device ISA/IL from the specific GPU model and vendor and of the source code from a specific programming language. Moreover, working on LLVM allows to extend the analysis to include the CPU as a scheduling target.

Since the final target of our research is to partition and to schedule generic algorithms in a control-flow-aware manner, the most important step consists in developing a partitioning and scheduling system based on code similarity pattern discovery engine to recognize well-known parallel patterns inside generic algorithms. The engine is paired with a database of popular computations, such as map, reduce, scan, convolution, matrix reduction and multiplication. The classification metric defined in the previous steps is used to train the scheduler on a

Table 1 Specifications of the testing platform.

Device	Clock rate	SIMDs	Cores	Processing power
AMD A8-3850 (CPU)	2.9 GHz	-	4 CPUs	-
AMD 6550D (Integrated)	600 MHz	5	400 radeon	480 GFLOPS
AMD HD 5870 (Discrete)	850 MHz	20	1600 radeon	2720 GFLOPS

particular system to determine the integrated versus discrete GPU convenience threshold. For each computation in the database we store a marker representing the device on which it is more suitable to execute. Finally, the database and the markers are employed to partition a generic algorithm in terms of the patterns it contains, scheduling each pattern recognized on the device stored in the relative marker. For this part of the work we can benefit of many researches on code similarity and pattern discovery [16, 17, 18].

3 Preliminary results

In order to investigate on the chance for the CPU, the integrated and the discrete GPU to be specialized to accelerate different sets of computations, we start analyzing the performances of integrated and discrete GPUs in running a set of algorithms with specific computing requirements. The algorithms that compose the test suite are vector/matrix addition (saxpy), reduction, convolution and matrix multiplication. The choice of the set of algorithms has been driven by the aim to take into account both memory-bound and compute-bound algorithms.

Each algorithm is executed on the testing platform (table 1) under different conditions. In particular, we run each algorithm using all the possible data-transfer strategies, like mapping, placement (ALLOC_HOST_PTR, USE_PERSISTENT_MEM_AMD, etc.) and pre-pinning and employing different data types (float, float2, float4, etc.). For each algorithm and for each device we select the conditions that lead to the lowest completion time.

3.1 Experimental results

Figures 1, 2 and 3 compare the completion times of the integrated GPU and of the discrete GPU resulting from the execution of the benchmarks. Since we show the ratio between the completion times of the two devices, values higher than one mean that the integrated GPU is faster than the discrete GPU, while a ratio lower than one signifies that the discrete GPU is faster than the integrated one.

The integrated GPU is more efficient in executing both saxpy and reduction for every input size. Matrix multiplication falls into the opposite situation, since the discrete GPU outperforms the integrated one regardless the matrix size. Finally, convolution exhibits a mixed behaviour, where the discrete GPU is faster for small input matrixes and gradually becomes slower than the integrated GPU as bigger the matrix size. Since the convolution algorithm depends on both the input matrix size and the filter size, we also run the algorithm fixing the input matrix size and gradually increasing the filter. The results of this test, shown in figure 4, confirm the partial convenience of the integrated GPU and allow to conclude that the discrete GPU outperforms the integrated one for small input matrixes and for big filters.

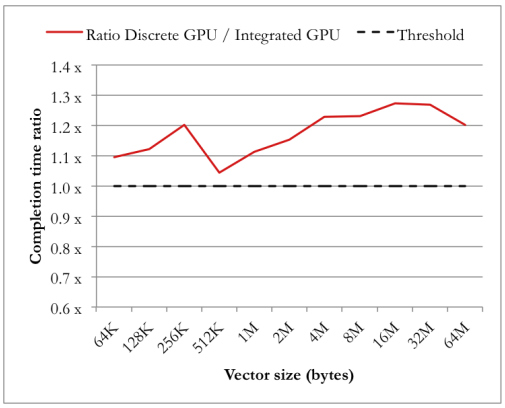


Figure 1 Saxpy completion time ratio: discrete GPU / integrated GPU.

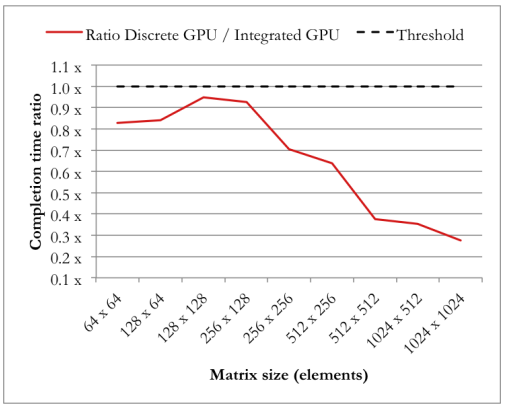


Figure 2 Matrix multiplication completion time ratio: discrete / integrated GPU.

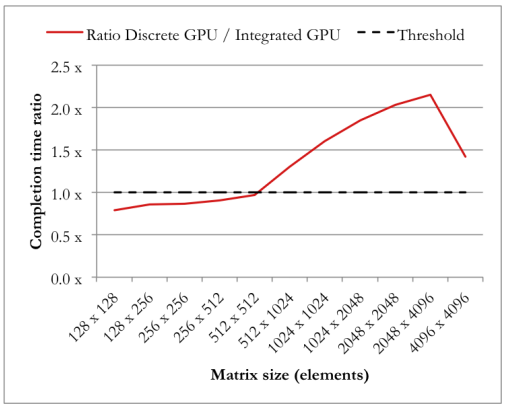


Figure 3 Convolution completion time ratio: discrete / integrated GPU.

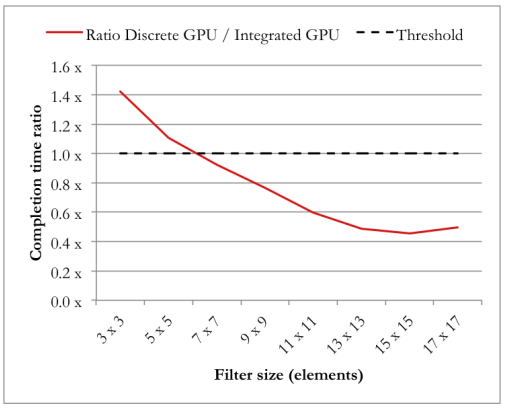


Figure 4 Convolution completion time ratio varying filter size: discrete / integrated GPU.

3.2 Computation Density

For each algorithm, we calculate the ratio between the number of operations performed on data and the amount of data transferred between the CPU and the GPU. We call this metric *Computation Density*(CD). The aim of this metric is to classify the efficiency of integrated versus discrete GPUs in executing a particular algorithm. As shown in table 2, saxpy and reduction have a constant CD while matrix multiplication is characterized by a CD that is linear on the data size. In matrix convolution, the CD depends both on the input matrix size and on the filter size. When the matrix size is much bigger than the filter area ($N \gg M^2$), the CD can be approximated to the following, which is constant on the input matrix size.

$$CD_{smallfilter} = 2M^2 \tag{1}$$

When the matrix size and the filter area are similar ($N \approxeq M^2$), the ratio can be instead approximated using the below formula, which is linear on the input size.

$$CD_{bigfilter} = \frac{2N^3}{2N^2} = N \tag{2}$$

The values of CD calculated for the algorithms taken into account suggest that the efficiency of integrated GPUs versus discrete GPUs is highly correlated to the balance

between the amount of operations performed on data and the amount of data transferred to and from the GPU. In particular, when the CD is constant and below a certain threshold, the integrated GPU is the faster device. Whereas the CD is more than constant on the input size (e.g. linear, for matrix multiplication), the discrete GPU is instead capable of outclassing the integrated GPU. The most interesting test revealing this correlation is the matrix convolution, where CD can be considered constant on the input size except for matrixes and filters of similar sizes. Since the filter size is always smaller than the input matrix size, there are two chances for input matrix and filter to have similar sizes, that is decreasing the input matrix size and increasing the filter size. These two situations are the one discovered executing the algorithm on the testing platform (fig. 3 and 4).

Table 2 Computation Density of the tested algorithms

Saxpy	$\frac{N^2}{3N^2} = \frac{1}{3}$ (N^2 matrix size)
Reduction	$\frac{N}{N+1} \approx 1$ (N vector size)
Matrix multiplication	$\frac{2N^3}{3N^2} = \frac{2}{3}N$ (N^2 matrix size)
Matrix convolution	$\frac{2M^2N^2}{2N^2+M^2}$ (N^2 matrix size, M^2 filter size)

4 Conclusion

In this paper we shown that APU's GPUs and discrete GPUs can effectively accelerate different kind of computations, giving us the chance to specialize algorithms in order to obatain the best performances from an hybrid multi-GPU system. We also tried to correlate the test results with the characteristics of each specific algorithm, leading to a metric called *Computation Density*. Actually, many aspects can influence the completion time, such as thread synchronization, memory access patterns (influencing bank conflicts and coalescing) and loop unrolling. We are working to refine the metric to take into account all these aspects while trying to keep it as simple as possible. The following step is to take into account the CPU, which is particularly challenging due to it's different execution model and to the CPU-GPU memory sharing. In particular, memory sharing poses contention problems, since empoying the CPU to perform part of a computation may limit the memory access bandwidth and therefore the performances of the integrated GPU. Finally, we plan to use the Computation Density[1] to train a classifier on a large set of widely used computational patterns. The target is to employ machine-learning and parallel patterns discovery to partition and classify general purpose algorithms on the basis of the set of parallel computational patterns recognized during code analysis.

References

1 M. Daga, A. M. Aji, and W.-c. Feng. On the efficacy of a fused cpu+gpu processor (or apu) for parallel computing. In *Proceedings of the 2011 Symposium on Application Accelerators in High-Performance Computing*, SAAHPC '11, pages 141–149, Washington, DC, USA, 2011. IEEE Computer Society.

2 A. D. Malony, S. Biersdorff, S. Shende, H. Jagode, S. Tomov, G. Juckeland, R. Dietrich, D. Poole, and C. Lamb. Parallel performance measurement of heterogeneous parallel sys-

[1] Refined and eventually intergated with other metrics

tems with gpus. In *Proceedings of the 2011 International Conference on Parallel Processing*, ICPP '11, pages 176–185, Washington, DC, USA, 2011. IEEE Computer Society.

3 S. Ryoo, C. I. Rodrigues, S. S. Stone, J. A. Stratton, S.-Z. Ueng, S. S. Baghsorkhi, and W.-m. W. Hwu. Program optimization carving for gpu computing. *J. Parallel Distrib. Comput.*, 68:1389–1401, October 2008.

4 R. Vuduc, A. Chandramowlishwaran, J. Choi, M. Guney, and A. Shringarpure. On the limits of gpu acceleration. In *Proceedings of the 2nd USENIX conference on Hot topics in parallelism*, HotPar'10, pages 13–13, Berkeley, CA, USA, 2010. USENIX Association.

5 The Tradeoffs of Fused Memory Hierarchies in Heterogeneous Computing Architectures. K. Spafford, J. S. Meredith, S. Lee, D. Li, P. C. Roth and J. S. Vetter. *Future Technologies Group, Computer Science and Mathematics Division, Oak Ridge National Laboratory, 1 Bethel Valley Road-MS6173, Oak Ridge, TN 37831.*

6 G. de Bailliencourt. M-JPEG Decoding Using OpenCL on Fusion *AMD Fusion Developer Summit*, 2011.

7 Accelerating Lossless Data Compression with GPUs. R.L. Cloud, M.L. Curry, H.L. Ward, A. Skjellum and P. Bangalore. *July, 2011.*

8 S. Keely. Heterogeneous Kd-tree Construction on an APU *AMD Fusion Developer Summit*, 2012.

9 S. Venkatasubramanian, R. W. Vuduc. Tuned and Wildly Asynchronous Stencil Kernels for Hybrid CPU/GPU Systems. *Georgia Institute of Technology, College of Computing, School of Computer Science, 266 Ferst Drive, Altanta, Georgia, USA.*

10 D. Grewe and M.F.P. O'Boyle A Static Task Partitioning Approach for Heterogeneous Systems Using OpenCL. *School of Informatics, The University of Edinburgh, UK*, 2011.

11 C. Luk [1], and S.H.H. Kim [2]. Exploiting Parallelism on Heterogeneous Multiprocessors with Adaptive Mapping. [1] *SSG Software Pathfinding and Innovation, Intel Corporation, Hudson, MA. School of Computer Science.* [2] *Georgia Institute of Technology, Atlanta, GA.*

12 Multi-GPU and Multi-CPU Parallelization for Interactive Physics Simulations. E. Hermann[1], B. Ran[1], F. Faure[2]„ T. Gautier[1] and J. Allard[1]. [1]*INRIA.* [2]*Grenoble University.*

13 M. D. Linderman, J. D. Collins, H. Wang and T. H. Meng. Merge: A Programming Model for Heterogeneous Multi-core Systems Abstract. *2011.*

14 V.J. Jimenez[1], L. Vilanova[2], I. Gelado[2], M. Gil[2], G. Fursin[3] and N. Navarro[2]. Predictive Runtime Code Scheduling for Heterogeneous Architectures. [1]*Barcelona Supercomputing Center (BSC).* [2]*Departament d'Arquitectura de Computadors (UPC).* [3]*ALCHEMY Group, INRIA Futurs and LRI, Paris-Sud University.*

15 Advanced Micro Device. *AMD Accelerated Parallel Processing with OpenCL. Revision 2.2. June, 2012.*

16 M. Miron Bernal, H. Coyote Estrada, J. Figueroa Nazuno. Code Similarity on High Level Programs. *Proceedings of the 18th Autumn Meeting on Communications, Computers, Electronics and Industrial Exposition. (IEEE - ROCC07). Acapulco, Guerrero, Mexico. 2007.*

17 N. Wu, S. M. M. Tahaghoghi. Evolving similarity functions for code plagiarism detection. *Honours Thesis. School of Computer Science and Information Technology. RMIT University. Melbourne, Australia. October, 2007.*

18 J. Dong[1], Y. Sun[2], Y. Zhao[1]. Design Pattern Detection by Template Matching. [1]*Computer Science Department, University of Texas, TX 75083, USA.* [2]*American Airlines, 4333 Amon Carter Blvd, Fort Worth, TX 76155, USA.*

19 J. Holewinski. PTX Back-End: GPU Programming with LLVM. *The Ohio State University. LLVM Developer's Meeting. November 18, 2011.*

Abstracting Continuous Nonpolynomial Dynamical Systems

William Denman

Computer Laboratory, University of Cambridge, UK
william.denman@cl.cam.ac.uk

Abstract

The reachability problem, whether some unsafe state can be reached, is known to be undecidable for nonlinear dynamical systems. However, finite-state abstractions have successfully been used for safety verification. This paper presents a method for automatically abstracting nonpolynomial systems that do not have analytical or closed form solutions.

The abstraction is constructed by splitting up the state-space using nonpolynomial Lyapunov functions. These functions place guarantees on the behaviour of the system without requiring the explicit calculation of trajectories. MetiTarski, an automated theorem prover for special functions (sin, cos, sqrt, exp) is used to identify possible transitions between the abstract states. The resulting finite-state system is perfectly suited for verification by a model checker.

1998 ACM Subject Classification I.6.4 Model Validation and Analysis

Keywords and phrases Formal Verification, Automated Theorem Proving, Abstraction, Nonpolynomial System, MetiTarski

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.42

1 Introduction

Abstracting continuous systems into a finite state representation has been a successful method for the formal verification of real world problems. Current abstraction methods can only handle linear or polynomial nonlinear systems. Nonpolynomial terms must be either linearised or over-approximated [4]. These approximations can introduce abstract states that are seen as false-positives by the model checker. In this paper, the automated theorem prover MetiTarski is used to create an abstraction of a nonpolynomial continuous system by working with the nonpolynomial terms directly. This goal is to enhance the quality of the resulting finite state abstraction.

MetiTarski [2] is an automated theorem prover for arithmetical conjectures involving transcendental functions (sin, cos, exp etc.). It has been successful in proving arithmetical theorems that are used to verify analogue circuits [3] and linear hybrid systems [1].

Most systems of interest can only be specified using nonlinear differential equations. This is because, not surprisingly, nonlinear systems present a richer set of dynamics. It is for these reasons that both qualitative analysis and repeated numerical simulation is used [10]. The finite level of precision of numerical methods is often a source of significant error.

Safety, the fact that some bad behaviour will never happen, is the most important property that should be verified for a system. The reachability computation remains the most common way to check safety of a system. Unfortunately, the reachability decision problem of continuous systems is undecidable [6]. Abstraction methods are commonly employed to solve this problem.

By abstracting properly and preserving the relevant underlying behaviour of the system, tools that are already developed can be used. Sloth and Wisniewski [12] developed a method

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 42–48

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

for creating a sound and complete abstraction of continuous systems using Lyapunov functions. By using the Lyapunov function as a predicate for partitioning, they were able to convert the infinite state space of a continuous system into timed automata. They are however only able to abstract a restricted class of linear systems.

Another abstraction method borrows ideas from the domain of qualitative reasoning. Qualitative reasoning is motivated by the idea that numerical simulation is limited when not all the parameters of the system are known. Instead of trying to compute a solution, it is sufficient to look at how the vector field itself changes over time. Tiwari [14] uses predicates that evaluate over the three symbols $\{+, -, 0\}$ to split up the infinite state space. This construction of the abstraction uses the decidability of the first order theory of real closed fields [13] to compute the transitions between abstract states. Once the abstraction is created then a model checker is used to evaluate Computation Tree Logic (CTL) properties on the abstract system. The method proposed by Tiwari is limited to nonlinear polynomial vector fields.

2 Dynamical Systems

A dynamical system can be thought of as an abstract entity that changes its behaviour and state with respect to time. The *state* is the current value of the variables of the system. The *behaviour* is a function that returns the next state of the system, given the current state. These two quantities are required to completely model the system.

▶ **Definition 1** (Dynamical System)**.** An n-dimensional dynamical system DS is represented by the state vector $\mathbf{x}(t) \in \mathcal{R}^n$ and a function $f : \mathcal{R}^n \to \mathcal{R}^n$

For continuous systems time will progress as a smooth function. Instead of giving an explicit value of the next state, function f will define how the system evolves continuously. The simplest way to model this smooth change of variables is using *differential* equations.

▶ **Definition 2** (Continuous Dynamical System)**.** A continuous dynamical system is compactly modelled using a set of differential equations of the form

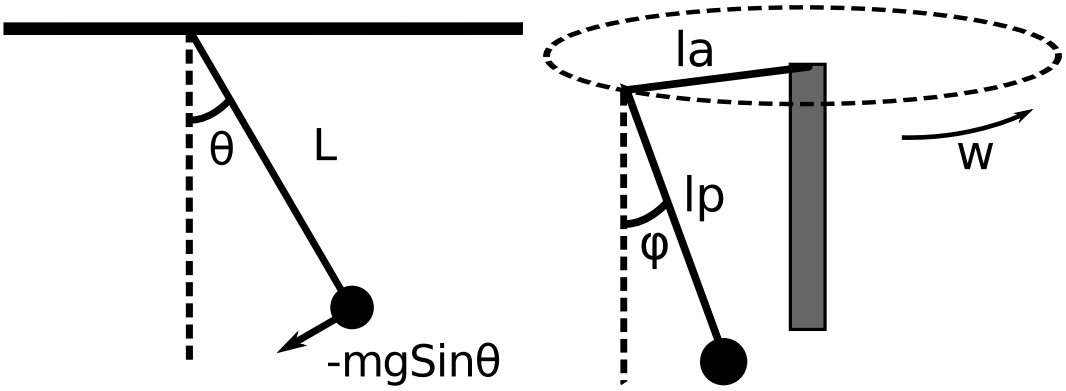
$$\mathbf{x}'(t) = f(\mathbf{x}(t)) \tag{1}$$

It is common convention to drop the explicit reference to the time variable.

$$\mathbf{x}' = f(\mathbf{x}) \tag{2}$$

The benefit of using differential equations is that continuous systems can be completely represented by how their variables change. Even for the simplest of systems, it can be quite difficult and in most cases impossible to analytically solve Equation (1). If the functions $f(x)$ are polynomial, that is $f(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0$ where n is a non-negative integer and $a_0, a_, .., a_n$ are constants then the resulting system is polynomial. Otherwise the system is nonpolynomial.

▶ **Example 1** (The Pendulum)**.** Take for instance the friction-free pendulum of Figure 1a. A rod of length L is attached to a ball of mass m. As the ball swings, the angle θ between the rod and the vertical changes. The angular velocity (rotational speed in the tangential direction) $\omega(t)$ is equivalent to the change of the angle θ or $\frac{d\theta}{dt}$. Acceleration, velocity and position of the ball are related by $a = v' = x''$. The arc-distance travelled by the ball is $x = \theta L$. The effective force returning the ball to the center is $mg \sin\theta$. The differential



(a) Regular pendulum **(b)** Whirling pendulum

Figure 1 Two nonpolynomial systems.

equations of the system can be derived from Newton's 2nd Law $F = ma$. Taking $\frac{F}{m} = a$, $a = x'' = (\theta L)'' = \omega' L$ gives the system in state space form

$$\theta' = \omega \tag{3a}$$

$$\omega' = -\frac{g}{L}\sin\theta \tag{3b}$$

This model is exactly described by two simple differential equations. The problem is that there is no known method that can obtain a solution with respect to time for either of the state variables ($\theta(t)$ or $\omega(t)$). This is due to the nonpolynomial $\sin\theta$ term in Equation (3b). Under certain conditions, nonpolynomial systems like Example 1 can be approximated by a *linear* system that is guaranteed to have a closed form solution.

▶ **Definition 3** (Linear System)**.** When $f(\mathbf{x})$ (Equation (2)) is defined by an affine line $ax + b$, the continuous system is said to be linear. If the state vector $\mathbf{x}$ is n-dimensional then, $f(\mathbf{x}) = Ax + b$ where A is an n by n matrix b is an n vector.

Since the linear system is defined using a square matrix, it is easy to extract the eigenvalues [11]. These eigenvalues can be used to construct the solution to the system of equations and to understand the qualitative behaviour of the system's trajectories. The interesting result is that if a nonpolynomial system is replaced by a linear approximation, the eigenvalues of the approximation can be used to understand the behaviour of the original system. The linear approximation is only valid in a close neighbourhood of a particular point. In dynamical system analysis, this is usually chosen to be an *equilibrium point*.

▶ **Definition 4** (Equilibrium Point)**.** An equilibrium or fixed point is a location in the state-space $\tilde{\mathbf{x}}$ where $f(\tilde{\mathbf{x}}) = \tilde{\mathbf{x}}$. When the system is at the equilibrium point, it will stay there for all time if not disturbed. If a slight disturbance causes the system to leave the equilibrium point and never return then the equilibrium point is *unstable*, otherwise the system is *stable*.

Since nonpolynomial systems cannot be solved analytically, verification relies on the analysis of the qualitative behaviour of the system near its equilibrium points. This *qualitative analysis* looks to see how sets of trajectories move in the state-space. For linear systems, if the eigenvalues of the system all have a negative real part, then the stability of the equilibrium point is guaranteed. If the real parts of the eigenvalues are all 0 then nothing can be concluded and the linearisation method fails. An alternative method that operates on the original nonlinear vector field directly can be used instead.

▶ **Definition 5** (Lyapunov Function)**.** A function $V(x)$ is a Lyapunov function, if for an equilibrium point (fixpoint) located at the origin (0,0) the following conditions hold

$$V(x) > 0 \qquad \text{for } x \neq 0 \tag{4a}$$

$$V(0) = 0 \tag{4b}$$

$$\frac{\partial V(x)}{\partial t} \leq 0 \qquad \text{for all } x \tag{4c}$$

If a Lyapunov function exists, then the equilibrium point is guaranteed to be stable [10]. The Lyapunov property is a sufficient condition for stability.

Return to Example 1 but assume now that the system is real by including friction effects. $V(x)$ can be chosen as the total energy (kinetic plus potential) of the system. It is clear that when the pendulum is displaced, energy is put into the system causing $V(x)$ to increase and $V(x) > 0$. The energy of the system will only be zero when the pendulum has stopped swinging and is hanging straight down at position 0, therefore $V(0) = 0$. The system continuously loses energy due to friction and $V(x)$ is always decreasing, implying that $V'(x) < 0$. Since the three constraints have been met, $V(x)$ is an Lyapunov function and by definition the equilibrium point at rest is stable.

The Lyapunov method does not require that $V(x)$ be the energy of the system, any function can be used. The caveat is that finding a Lyapunov function in general can be quite difficult. There are several advanced methods based on sum-of-squares (SOS) techniques that make the search for the Lyapunov function tractable. These methods have been implemented in a MATLAB package called SOSTOOLS [9]. The next section describes an abstraction algorithm that uses Lyapunov functions.

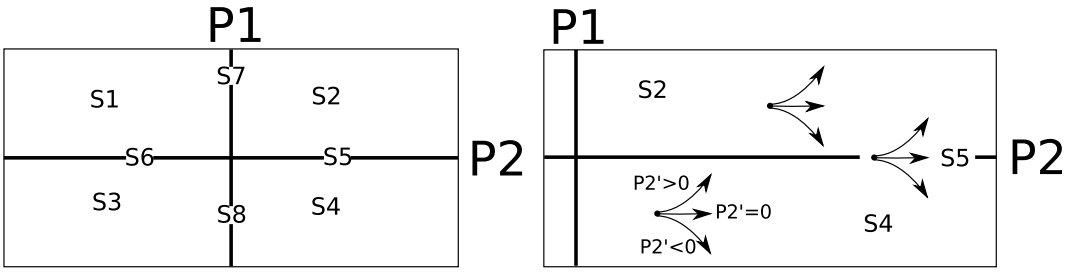
3 Abstracting the Dynamical System

The end-goal of verification is to prove that a system has been built correctly. For continuous systems we specifically want to prove that all trajectories starting in a *safe* state will never reach a *bad* or *unsafe* state. This reachability analysis is known to be decidable for discrete systems such as finite automata, but it is undecidable for nonpolynomial systems. Abstraction methods are a shortcut used to obtain a decidability result from the undecidable brick wall.

The abstraction method of Tiwari [14] discretizes the state-space using predicates evaluated over three symbols $\{+, -, 0\}$. Each abstract state is defined by a conjunction of these predicates. For the example in Figure 2a, predicates P1 and P2 represented by thick lines have been used to discretize a two dimensional state space. Taking P1 : $f(x, y) = x$ and P2 : $f(x, y) = y$, state S1 is $P1 > 0 \wedge P2 > 0$, S7 is $P1 = 0 \wedge P2 > 0$ and so on.

The difficulty in creating finite state abstractions is choosing the predicates used to discretize the state space. Tiwari has developed several heuristics for defining *good* predicates. Lyapunov functions are a good choice because they represent a positively invariant set. By definition, the solutions of the system will only pass through the level sets of Lyapunov functions in one direction. Including Lyapunov functions greatly simplifies the construction of abstract transition relations by limiting the reachable state space.

A decision procedure is used to determine all possible transitions between the abstract states. For instance, in the example shown in Figure 2a the following cases must be checked to determine the transitions between S2, S4 and S5. For brevity, P1 is assumed to be positive. Transitions between abstract states are decided by checking the sign of the derivative of the predicate with respect to the vector field of the system. To take this derivative, the chain rule for partial derivatives is used $\frac{dPn}{dt} = \frac{\partial Pn}{\partial x}\frac{dx}{dt} + \frac{\partial Pn}{\partial y}\frac{dy}{dt}$.



(a) Discretizing the state space with predicates **(b)** Transitioning between abstract states

Figure 2 Tiwari's Abstraction Method.

1. If the current state is S4 : (P2 < 0) then:
   - If $P2' > 0$, the next state is either S4 or S5
   - If $P2' = 0$, the next state is S4
   - If $P2' < 0$, the next state is S4
   - If unknown, the next state is S4 or S5
2. If the current state is S5 : (P2 $= 0$) then:
   - If $P2' > 0$, the next state is S2
   - If $P2' = 0$, the next state is S5
   - If $P2' < 0$, the next state is S4
   - If unknown, the next state is S2 or S5 or S4
3. If the current state is S2 : (P2 > 0) then:
   - If $P2' > 0$, the next state is S2
   - If $P2' = 0$, the next state is S2
   - If $P2' < 0$, the next state is S5 or S2
   - If unknown, the next state is S2 or S5

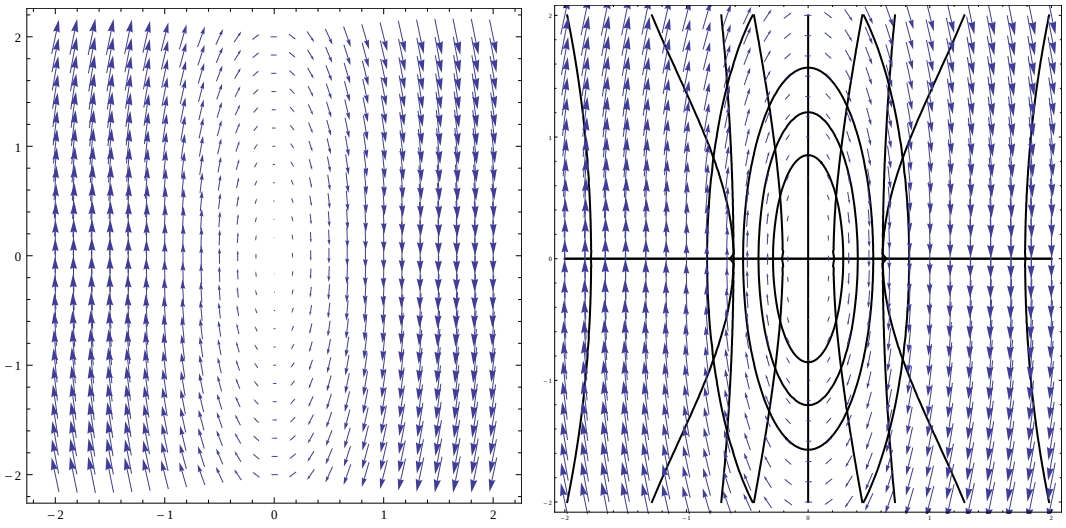
One issue is that the decision procedure used by Tiwari is only applicable to polynomials. This restriction limits the type of systems that can be analysed. The next example shows how Tiwari's method is extended to work with nonpolynomial systems. MetiTarski is used to reason about the inequalities that are generated during the abstract transition analysis described above.

▶ **Example 2** (Whirling Pendulum)**.** Consider Figure 1b, where a pendulum of length l_p is attached to a movable rigid arm of length l_a. Taking the following assumptions: the pendulum is light enough to be swung up with a small ϕ', ignore friction and consider that each pendulum arm is thin enough to make the moment of inertia negligible [5]. With $x_1 = \phi$ and $x_2 = \phi'$ the system of equations are

$$x_1' = x_2 \tag{5a}$$

$$x_2' = \omega^2 \sin x_1 \cos x_1 - \frac{g}{l_p} \sin x_1 \tag{5b}$$

The predicates used to split up the state space are obtained by repeatedly taking the



(a) Vector field **(b)** Discretized state-space

Figure 3 The whirling pendulum system.

derivative of the vector field.

$$P1 = x_2'' = -x_2 \sin^2(x_1) + x_2 \cos^2(x_1) - 10x_2 \cos(x_1) \tag{6a}$$

$$\begin{aligned} P2 = P1' = \frac{1}{4} \sin(x_1)(8{x_2}^4(8\cos(x_1) - 5) - 8{x_2}^2(561\cos(x_1) - 210\cos(2x_1) \\ + 11\cos(3x_1) - 65) - 1104\cos(x_1) - 8040\cos(2x_1) + 2304\cos(3x_1) \\ - 175\cos(4x_1) + 4\cos(5x_1) + 4095) \end{aligned} \tag{6b}$$

and a Lyapunov function of the system is found using SOSTOOLS

$$V1 = 0.3345x_2^2 + 1.4615\sin^2 x_1 + 1.7959\cos^2 x_2 - 6.689x_2 + 4.8931 \tag{7}$$

The behaviour of the system is shown in the vector field plot of Figure 3a with x_1 on the horizontal axis and x_2 on the vertical axis. Using the predicates obtained from the equations of the system (Equations 6a and 6b) along with several level sets of the Lyapunov function (Equation 7 : $V1 = 0.25, V1 = 0.5, V1 = 0.85$ and $V1 = 2$), the state-space is discretized as shown in Figure 3b. MetiTarski is used to determine the transitions between the abstract states using the method described in Section 3. The methods of Sloth, Wisniewski and Tiwari cannot deal with this system because of the nonpolynomial components.

4 Conclusion

An abstract system has been constructed by choosing the appropriate predicates and discretizing the continuous state space of a nonpolynomial dynamical system. The abstract transitions have been automatically obtained using MetiTarski. The resulting finite state transition system can be sent to a model checker for verification purposes.

One important open question is concerned with choosing good predicates. Tiwari's method uses predicates that are constructed by taking repeated derivatives of the vector fields. The motivation being that for polynomial systems this process terminates. For nonpolynomial systems this is not necessarily the case. It will be necessary to quantify the *quality* of the

abstractions. One potential option to increase the quality of the generated abstractions is to use the Counter Example Guided Abstraction Refinement (CEGAR) framework.

Barrier Certificates can be used for safety analysis [7] and are another source of good predicates. Instead of being concerned with the stability of an equilibrium point, they are used to prove that certain states of a system cannot be reached. This is done using Lyapunov theory (see Definition 5). The important point is that SOSTOOLS can be used to search for Barrier Certificates. Linear hybrid systems have been successfully verified using these techniques [8]. Future work includes using MetiTarski for determining abstract transitions of systems discretized by Barrier Certificates.

References

1 B. Akbarpour and L. C. Paulson. Applications of MetiTarski in the verification of control and hybrid systems. In *Hybrid Systems: Computation and Control*, volume 5469 of *Lecture Notes in Computer Science*, pages 1–15, 2009.

2 Behzad Akbarpour and Lawrence C. Paulson. MetiTarski: An automatic theorem prover for real-valued special functions. *Journal of Automated Reasoning*, 44:175–205, 2010.

3 W. Denman, B. Akbarpour, S. Tahar, M. H. Zaki, and L. C. Paulson. Formal verification of analog designs using MetiTarski. In *Formal Methods in Computer-Aided Design. FMCAD 2009*, pages 93 –100, November 2009.

4 Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV)*, LNCS. Springer, 2011.

5 K Furuta, M Yamakita, and S Kobayashi. Swing-up control of inverted pendulum using pseudo-state feedback. *Part I: Journal of Systems and Control Engineering*, 206:263–269, 1992.

6 Emmanuel Hainry. Reachability in linear dynamical systems. In *Proceedings of the 4th conference on Computability in Europe: Logic and Theory of Algorithms*, CiE '08, pages 241–250, Berlin, Heidelberg, 2008. Springer-Verlag.

7 Stephen Prajna. Barrier certificates for nonlinear model validation. *Automatica*, 42(1):117–126, 2006.

8 Stephen Prajna and Ali Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *In Hybrid Systems: Computation and Control*, pages 477–492. Springer, 2004.

9 Stephen Prajna, Antonis Papachristodoulou, Peter Seiler, and Pablo A. Parrilo. SOSTOOLS: Sum of squares optimization toolbox for MATLAB, 2004.

10 Shankar Sastry. *Nonlinear Systems*. Springer, 1999.

11 Edward R. Scheinerman. *Invitation to Dynamical systems*. Dover Publications, 1996.

12 C. Sloth and R. Wisniewski. Abstraction of continuous dynamical systems utilizing lyapunov functions. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 3760–3765, December 2010.

13 Alfred Tarski. A decision method for elementary algebra and geometry. Technical report, RAND Corp., 1948.

14 Ashish Tiwari and Gaurav Khanna. Series of abstractions for hybrid automata. In *Hybrid Systems: Computation and Control HSCC*, volume 2289 of *LNCS*, pages 465–478. Springer, March 2002.

Improving the Quality of Distributed Composite Service Applications

Dionysios Efstathiou[1], Peter McBurney[1], Noël Plouzeau[2], and Steffen Zschaler[1]

1 **Department of Informatics, King's College London**
{dionysios.efstathiou, peter.mcburney, steffen.zschaler}@kcl.ac.uk
2 **IRISA, University of Rennes 1**
noel.plouzeau@irisa.fr

Abstract

Dynamic service composition promotes the on-the-fly creation of value-added applications by combining services. Large scale, dynamic distributed applications, like those in the pervasive computing domain, pose many obstacles to service composition such as mobility, and resource availability. In such environments, a huge number of possible composition configurations may provide the same functionality, but only some of those may exhibit the desirable non-functional qualities (e.g. low battery consumption and response time) or satisfy users' preferences and constraints. The goal of a service composition optimiser is to scan the possible composition plans to detect these that are optimal in some sense (e.g. maximise availability or minimise data latency) with acceptable performance (e.g. relatively fast for the application domain). However, the majority of the proposed optimisation approaches for finding optimal composition plans, examine only the Quality of Service of each participated service in isolation without studying how the services are composed together within the composition. We argue that the consideration of multiple factors when searching for the optimal composition plans, such as which services are selected to participate in the composition, how these services are coordinated, communicate and interact within a composition, may improve the end-to-end quality of composite applications.

1998 ACM Subject Classification C.0 [General]: System architectures

Keywords and phrases Service Composition, Optimisation, Dynamism, Evolution

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.49

1 Introduction

Service-orientation promotes the creation of new value-added applications by composing pre-existing services regardless their location and platform technology [16]. *Service composition* [8, 9] is not only limited to functional composition, but also takes into account *non-functional* issues. Indeed, an application that does not obey the user's Quality of Service (QoS) constraints might be as useless as a service not providing the desired functionality.

Service composition in distributed environments faces four challenges: (a) a large space of possible deployment architectures are possible, (b) satisfaction of multiple users' (possibly conflicting) QoS preferences and constraints, (c) continuous system evolution and high dynamism since the components and their relationships are not known at design-time and may change at run-time, and (d) absence of an entity with global knowledge. These challenges demand a dynamic self-adaptive distributed solution [11, 17] for finding the composition architectures of high-quality that achieve the users' functional and non-functional goals.

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 49–55

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We argue that to achieve high-quality compositions, the composer should take into account not only the quality of each service in isolation, but also how services are coordinated, how they communicate and how they interact within the composition. Despite the vast research on the field of service composition, little is known about how these factors combined affect the aggregation of optimal compositions. The contention of this research is that the combined consideration of these factors may improve the overall quality of the composed applications.

The remainder of this paper is organised as follows. In Section 2 we present a motivating scenario for our research. Section 3 discusses some representative alternative approaches for optimising the process of service composition. Section 4 outlines our preliminary work for enabling the efficient on-the-fly creation of high-quality composite services. Finally, Section 5 concludes by outlining the plans of future work towards achieving our research goals.

2 Motivating Scenario

Our case study is a time-critical continuously evolving application: a fire-fighter tactical information and decision support system. The system's goal is to achieve improved knowledge of the emergency situation for better decision making via efficient sharing of real-time information between people in various hierarchical levels. For example, a commanding officer in an fire-fighting situation wants real-time information about on-field conditions, the availability of resources, and others, to make better decisions for risk identification, resource allocation, and others.

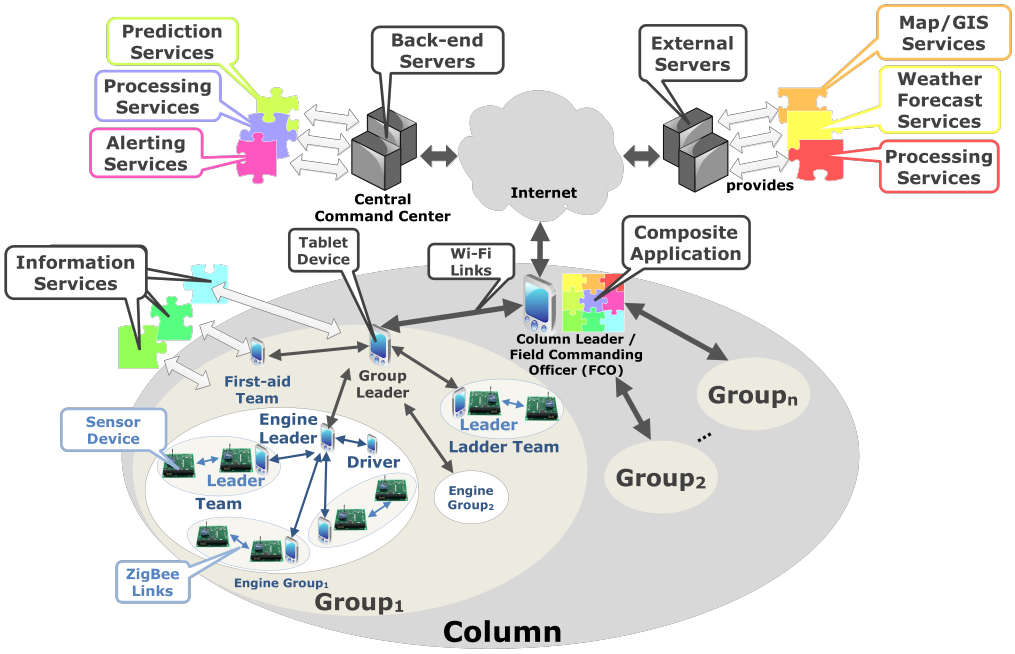


Figure 1 High-level architecture of the fire-fighting application scenario.

Figure 1 presents the groups emerging during an emergency operation which are (in order of hierarchy): team, engine group, group, and column. Each group has a leader, and each leader is equipped with a tablet. Each fire-fighter carries a number of special sensors (e.g. temperature, air quality, GPS module), and a number of sensors are deployed in the field to provide real-time data. The Field Commanding Officer (FCO), who is the highest ranked

person in the field (in our example the Column Leader), combines real-time information about the condition of the fire, local geography, fire evolution prediction services, available resources (e.g. personnel, water pumps), and others, to achieve better decision making. This application is composed of spatially distributed service components hosted on back-end servers and on-field mobile devices that cooperate to achieve a global goal.

Frequent changes of the composition architecture are necessary to deal with system dynamism (e.g. disconnections, battery depletion). Also, the system must continuously adapt to hierarchical and team changes, for example, due to initial danger underestimation there is a need for reinforcements which must be followed by corresponding adaptations (e.g. hierarchical changes, arrival of new services, etc.). Furthermore, the system must take into account that users of different hierarchical levels have different non-functional requirements. For example, the FCO demands to get information from the field in a fast way, while the Group Leader may require to have reliable messaging with all the members of his group.

For a given composite application, there may be many architectures that provide the same functionality, but with different levels of QoS. However, the parameters that influence the quality are not known before system execution. As a result, continuous reconfiguration is necessary to maintain the the high quality of the application during run-time.

3 Related Work

The related work of our research span several areas of the literature: (a) service composition models; (b) QoS-aware service composition; (c) self-adaptation; and (d) optimisation. We summarise the key lessons learned from our literature review that we will use to design our solution approach to the studied problem.

Service Composition Models. In inherently distributed environments, decentralised coordination of services may alleviate the problems (performance bottleneck, single-point-of-failure) [3] of traditional *service orchestration* [9], by enabling the distributed formation of compositions based on dynamic conditions (e.g. battery level, current bandwidth, etc.).

As services can be combined in unpredicted ways, there are many possible configurations to decentralise the coordination of a composite service which raises the question of what is the best way to achieve a decentralised coordination which satisfies a set of non-functional goals. Many researchers studied the problem of how to distribute service orchestration [2, 3, 4, 7, 15]. The inherent distributed and dynamic nature of service-oriented systems indicates the need for an adaptive and automated technique which can dynamically switch between possible orchestration approaches based on run-time conditions.

QoS-Aware Service Composition. In the current literature, optimal service composition is synonymous with the process of optimal QoS-aware service selection [5, 10, 12, 18]. In this problem, the goal is to find the combination of services that offers the required functionality, respects user's QoS requirements, and optimises the overall QoS of the composition. The main limitation of this family of approaches is that they only take into account the QoS of each service in isolation without studying how services are composed together. For instance, consider two services of very good quality (e.g. high availability, low response time). However, when it comes to composing them, the overall QoS of the composite application may be very low due to various reasons (e.g. slow/faulty communication link between interacting services or with the orchestrator that coordinates them).

Self-Adaptation. To realise a self-adaptive composition system, the following questions, as presented by Salehie and Tahvildari [14], need to be answered: (a) *where* the adaptation should occur; (b) *when* the adaptation action(s) should be applied; (c) *what* elements should

be affected; (d) *why* should adaptation be performed (adaptation goals); (e) *who* should be involved in the adaptation process; and (f) *how* the adaptation actions should be realised. The work of Cardellini *et al.* [6] is a first step in the right direction for answering the above questions in the context of dynamic service composition. We motivate the need for an autonomic controller with the goal to adjust the system's configuration on-the-fly based on the current conditions, because the configuration of the composition system is highly sensitive to the dynamic nature of the service-based environment.

Optimisation. Dynamic service composition can be seen as a dynamic multi-objective optimisation problem, where, given a set of services and a set of composition plans, the goal of the optimiser is to find the trade-off configurations that optimise the overall composition based on dynamic functional requirements and multiple, conflicting non-functional objectives. The optimisation algorithms used to solve this problem can be divided into two main categories: exact [12, 19] and approximative [5, 13], based on their precision (how close to optimal) and computational (how fast) complexity. Exact algorithms are able to produce solutions with guaranteed optimality but at exponential cost. On the other hand, approximative algorithms are able to produce sub-optimal solutions of "good" quality at polynomial time complexity. Some of the techniques applied to solve the studied problem are the following: linear and integer programming methods [12, 19], local search techniques [13], evolutionary metaheuristics [5], and others.

Composition Optimisation Frameworks Ardagna and Mirandola in [1] proposed a broker-based framework for run-time QoS-aware composition optimisation. However, their approach focuses only on the optimisation problem of QoS-aware service selection, ignoring the important issues of run-time adaptation, dynamic conditions, and distributed orchestration. Cardellini *et al.* [6] proposed a framework for run-time QoS-driven adaptation of SOA systems. However, their solution ignores the dynamic networking conditions for achieving the optimal composition. At the same time, they ignore the interaction and communication patterns for optimising the exchange of data between the various participating services. Finally, they only consider centralised orchestration, neglecting the fact that coordination of decentralised services may increase the overall composition performance.

4 Towards our Research Goal

Our goal is to supply a decision maker[1] with a set of trade-off composition configurations, and based on some problem-specific knowledge (e.g. QoS preferences, application context, etc.), to choose the optimal one.

As depicted in Figure 2, the composition system receives as input a set of abstract composition plans which describe abstractly the required functional tasks, a set of dynamic user functional and non-functional requirements, and a set of already deployed concrete services. Users insert requests into the composition engine, the front-end of the system. The composition engine identifies which concrete services implement the abstract tasks required by the service composition. Then, the optimisation engine tries to produce optimal composition configurations to realise user's goals. These configurations are provided to the decision maker agent who is responsible for optimising users' profit based on their dynamic preferences and constraints.

After a composite application is deployed, the monitoring component supervises the performance of the participating services and checks periodically whether it is necessary to

[1] An automatic agent whose goal is to maximise the user's profit.

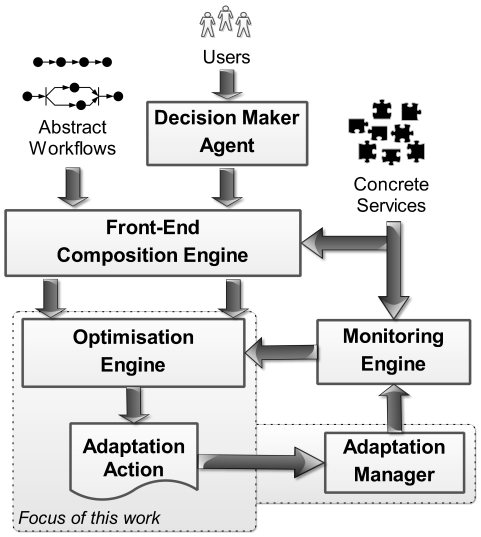


Figure 2 High-level Architecture and Focus of Our Research.

trigger a reconfiguration action by checking the quality of the composition. If an adaptation is triggered, the composition configuration will be dynamically updated according to the reconfiguration policy. The adaptation manager is responsible for implementing/applying the selected strategy. A simple adaptive behaviour of the composition system is to replace dynamically the bad performing services with better ones.

4.1 Scope of Research

The scope of our work is limited to finding, and maintaining a high-quality composition architecture based on users' dynamic preferences and real-time conditions. This means that other optimisation actions that may achieve the same goal, such as reconfiguring the amount of resources reserved for each service, replicating services, and others, are not considered in this study. Also, we do not focus on how to discover services in a distributed service environment. In our use-case scenario, this assumption can be justified by the fact that the descriptions (not the actual implementation) of the various services is known in advance.

4.2 Degrees of Freedom

The first step for optimising the composition process is to identify the most interesting degrees of freedom that enable us to change the provided quality of the composite applications without modifying their functionality. According to our literature review, we identified the following freedom variables: (a) how to decentralise the coordination of the service composition, (b) how to choose the communication patterns between interacting parties in a composition, and (c) how to select the concrete services for participating in the composition (see QoS-Aware Service Composition in Section 3).

As mentioned previously, there are many ways to decentralise the coordination of services in a distributed environment (e.g. how many distributed orchestrators to choose, where to place them, and others). Indeed, the way of realising the coordination of distributed services affects the overall quality of the composite application. Secondly, the various possible communication and interaction patterns between the participated services create opportunities for further improving the quality of the composition. For example, a Team

Leader A wants to send some data to his Group Leader. However, due to physical obstacles (e.g. in a building emergency scenario), the communication link between them is slow and faulty. Another Team Leader B passes nearby A that has a better connection with the Group Leader. Thus, it is better to forward A's data through B.

While there is a lot of work for service composition optimisation, to the best of our knowledge, there is no approach that takes into account simultaneously the above degrees of freedom for providing high-quality service compositions configurations at run-time.

4.3 Choosing an Optimisation Technique

The process of choosing the ideal technique is itself a multi-objective problem. Guided by our use-case scenario, we focus on approximate algorithms, and especially on the promising field of optimisation metaheuristics, such as Evolutionary Algorithms [20], that can provide "good enough" solutions in polynomial time, rather than solving the problem to true optimality.

5 Conclusion and Future Plans

Configuring composite applications of high quality that respect users' conflicting QoS objectives in the context of a continuously evolving distributed service-oriented environment, is a highly challenging research problem that requires deep investigation. Existing optimisation approaches focus only on the QoS of the participated services in isolation and ignore the existence of multiple factors that may affect the end-to-end quality of the composite application. In this paper, we proposed the consideration of multiple degrees of freedom when optimising the overall quality of a composite application. These degrees of freedom include the following: how services are selected to form a composition, how these distributed services are coordinated, how they communicate and interact with each other.

Further work needs to be done to establish whether the consideration of multiple factors leads to composite applications of improved quality. The next step towards achieving our goals is to design the meta-model that captures the main concepts of our service composition domain and abstracts from low-level details. The designed meta-model will enable the generation of model instances that represent possible composition plans. After generating the set of the possible plans (design space), our goal is to apply and compare various optimisation techniques (exact and approximate) to choose the more suitable approach. Finally, we aim at investigating the trade-off of executing the actual optimisation in a fully centralised way, hierarchical way, or totally distributed way among the network nodes.

References

1 Danilo Ardagna and Raffaela Mirandola. Per-Flow Optimal Service Selection for Web Services Based Processes. *Journal of Systems and Software*, 83(8):1512–1523, 2010.

2 Adam Barker, Jon B. Weissman, and Jano I. van Hemert. Eliminating The Middleman: Peer-to-Peer Dataflow. In *Proceedings of the 17th International Symposium on High Performance Distributed Computing*, pages 55–64, 2008.

3 Boualem Benatallah, Marlon Dumas, and Quan Z. Sheng. Facilitating the Rapid Development and Scalable Orchestration of Composite Web Services. *Distributed and Parallel Databases*, 17(1):5–37, 2005.

4 Walter Binder, Ion Constantinescu, and Boi Faltings. Decentralized Orchestration of Composite Web Services. In *International Conference on Web Services*, pages 869–876, 2006.

5 Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. An Approach for QoS-Aware Service Composition Based on Genetic Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1069–1075. ACM, 2005.

6 Valeria Cardellini, Emiliano Casalicchio, Vincenzo Grassi, Stefano Iannucci, Francesco Lo Presti, and Raffaela Mirandola. MOSES: A Framework for QoS Driven Runtime Adaptation of Service-Oriented Systems. *IEEE Transactions on Software Engineering*, 99, 2011.

7 Girish Chafle, Sunil Chandra, Vijay Mann, and Mangala Gowri Nanda. Orchestrating Composite Web Services Under Data Flow Constraints. In *IEEE International Conference on Web Services*, pages 211–218, 2005.

8 Anis Charfi and Mira Mezini. Hybrid Web Service Composition: Business Processes Meet Business Rules. In *Proceedings of the 2nd international conference on Service oriented computing*, ICSOC '04, pages 30–38. ACM, 2004.

9 Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

10 Michael C. Jaeger, Gero Mühl, and Sebastian Golze. QoS-Aware Composition of Web Services: An Evaluation of Selection Algorithms. In *OTM Conferences (1)*, pages 646–661, 2005.

11 Massimiliano Di Penta, Raffaele Esposito, Maria Luisa Villani, Roberto Codato, Massimiliano Colombo, and Elisabetta Di Nitto. WS Binder: a Framework to Enable Dynamic Binding of Composite Web Services. In *International Workshop on Service-oriented Software Engineering*, pages 74–80, 2006.

12 F. Rosenberg, P. Celikovic, A. Michlmayr, P. Leitner, and S. Dustdar. An End-to-End Approach for QoS-Aware Service Composition. In *IEEE International Enterprise Distributed Object Computing Conference*, pages 151–160, 2009.

13 Florian Rosenberg, Max Benjamin Müller, Philipp Leitner, Anton Michlmayr, Athman Bouguettaya, and Schahram Dustdar. Metaheuristic Optimization of Large-Scale QoS-aware Service Compositions. In *IEEE International Conference on Services Computing*, pages 97–104, 2010.

14 Mazeiar Salehie and Ladan Tahvildari. Self-Adaptive Software: Landscape and Research Challenges. *TAAS*, 4(2), 2009.

15 Quan Z. Sheng, Boualem Benatallah, Marlon Dumas, and Eileen Oi-Yan Mak. SELF-SERV: A Platform for Rapid Composition of Web Services in a Peer-to-Peer Environment. In *VLDB*, pages 1051–1054, 2002.

16 Latha Srinivasan and Jem Treadwell. An Overview of Service-Oriented Architecture, Web Services and Grid Computing. *HP Software Global Business Unit*, 2005.

17 Danny Weyns, Sam Malek, and Jesper Andersson. On Decentralized Self-Adaptation: Lessons from the Trenches and Challenges for the Future. In *Proceedings of the ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pages 84–93, 2010.

18 Tao Yu, Yue Zhang, and Kwei-Jay Lin. Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints. *ACM Transactions on the Web*, 1, 2007.

19 Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z. Sheng. Quality Driven Web Services Composition. In *Proceedings of the 12th International Conference on World Wide Web*, pages 411–421, 2003.

20 Eckart Zitzler, Marco Laumanns, and Stefan Bleuler. A Tutorial on Evolutionary Multiobjective Optimization. In *Metaheuristics for Multiobjective Optimisation*, pages 3–38, 2003.

Fine-Grained Opinion Mining as a Relation Classification Problem

Alexandru Lucian Gînscă

Alexandru Ioan Cuza University
Faculty of Computer Science, Iaşi
lucian.ginsca@info.uaic.ro

Abstract

The main focus of this paper is to investigate methods for opinion extraction at a more detailed level of granularity, retrieving not only the opinionated portion of text, but also the target of that expressed opinion. We describe a novel approach to fine-grained opinion mining that, after an initial lexicon based processing step, treats the problem of finding the opinion expressed towards an entity as a relation classification task. We detail a classification workflow that combines the initial lexicon based module with a broader classification part that involves two different models, one for relation classification and the other for sentiment polarity shift identification. We provided detailed descriptions of a series of classification experiments in which we use an original proximity based bag-of-words model. We also introduce a new use of syntactic features used together with a tree kernel for both the relation and sentiment polarity shift classification tasks.

1998 ACM Subject Classification I.2.7 Natural Language Processing

Keywords and phrases Opinion Mining, Opinion Target Identification, Syntactic Features

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.56

1 Introduction

Opinion mining is one of the applications of natural language processing with the biggest growth in recent years concerning the number of publications and dedicated conferences as well as industry applications. Most of them refer to opinion mining as a text classification task in which a text fragment is labeled as either positive or negative. In this paper, we focus on a more detailed approach of identifying the opinion expressed towards a certain target in a text fragment.

One of the basic and most used approaches for opinion mining is lexicon-based opinion generation and it has been used for opinion retrieval as a standalone method but there are also many research works that combined both lexicon based and text classification techniques in opinion mining systems [1]. The authors of [11] mention probabilistic models as methods that have also been used to retrieve and classify opinions from documents [3]. The probabilistic approach relies on probabilistic assumptions based on frequency of query terms [4]. Another method described in [11] is the language model approach that has also been used for opinion retrieval [12]. Most language models imply word level processing, sentence level processing and paragraph level processing, but the core of language models is the bag-of-words representation. The state of the art statistical methods are based on the observation that similar opinion words frequently appear together in a corpus, as detailed in [14]. If two words frequently appear together within the same context, they are likely to share the same polarity.

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 56–61

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2 Proposed approach

We present in this section a novel fine-grained opinion classification workflow that combines an initial lexicon based step with a broader classification part that involves two different models and we briefly describe a new proximity based bag-of-words model and the use of tree kernels for relation classification and polarity shift identification.

2.1 A novel mixed lexicon/machine learning classification workflow

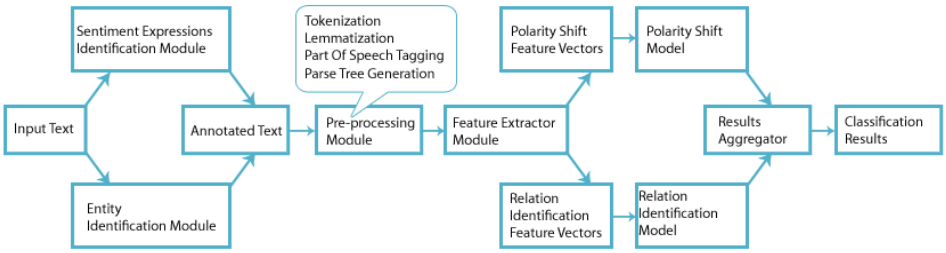


Figure 1 Classification workflow.

In Figure 1, we present an overview of our proposed classification workflow. The annotated text contains tags for entities and for sentiment bearing expressions. The actual entities and sentiments will be later replaced by an abstract token for the classification model as detailed in the following sections. The pre-processing module deals with the word level and sentence level text processing methods, such as lemmatization, part of speech tagging or generating a parse tree and depends on the features that will be later used in the workflow by the classifiers. The feature extraction module builds feature vectors both for the relation identification classifier as well as for the polarity shift identification one.

One of the key aspects of our approach is that we treat the fine-grained opinion identification problem as a relation identification one that is independent of the entity identification and sentiment extraction modules. This also represents one of the main advantages of the workflow that we propose, the fact that it can be easily adapted for different sentiment identification contexts, not only allowing different types of entities, but also different semantics of the expressed sentiments.

2.2 Relation identification with a novel proximity based bag-of-words model

The bag-of-words model is a common practice in text classification in which a document is represented by a vector of words. The vector is built from a dictionary that gathers all of the words from all the documents in the corpus. Three basic variations of the bag-of-words model can be identified: occurrence, where the values of the vector are 1 if the word appears in the document and 0, otherwise; appearances, where the values of the vector represent the number of times a word appear in that document and tf-idf, where the term frequency-inverse document frequency of the words of that document in respect with the whole corpus is used.

The main motivation for a different type of a bag-of-words model is the intrinsic nature of the classical model that does not take into consideration the position of the words in the sentence. Models that try to solve this problem by using n-grams (usually up to 5-grams) instead of unigrams have the problem of an exponential increase in feature space. This is

why we propose a different type of a bag-of-words model designed specifically for the binary relation identification problem that uses the proximity measured in number of tokens between words.

The model is built as follows: For each word in the dictionary that is found in the sentence, we first compute the number of tokens (words, punctuation) between the word and the SENTIMENT token and then the number of tokens between the word and the TARGET token. If the word appears in the sentence after the SENTIMENT token, the value that is put in the feature vector is the number of tokens between the word and the SENTIMENT token multiplied by -1. The same applies for the case in which the word is situated after the TARGET tag.

2.3 Relation identification with a tree kernel based model

The tree kernel is a function K(T1, T2) that returns a normalized similarity score in the range (0,1) for two trees T1 and T2 [2]. Details regarding the formal definition and in depth descriptions of tree kernels can be found in [16].

For the task of relation identification in the context of fine-grained opinion mining, we used Alessandro Moschitti's implementation of tree kernels that is described in [10] and [9] and is based on the SVM-Light library [5]. The SVM-Light implementation takes as input a parse tree with the binary label, but it also allows a combination of parse trees and numerical feature vectors for which the RBF or polynomial kernels can be used. It also allows the user to explicitly specify the way in which the results from each kernel are combined (addition or multiplication) and what weight is given to each kernel.

2.4 Opinion polarity shift identification with a tree kernel based model

Besides correctly identifying which sentiment bearing expression influences which target in a sentence, we are also interested to find out when a polarity shift for a sentiment expression that influences an entity takes place. A polarity shift is usually associated with negation and it represents the case in which the context changes a positive sentiment expression into a negative one and vice-versa. For the problem of polarity shift identification, we used a similar approach as for the relation identification one. For this task, we consider a positive instance, the case in which a polarity shift does not occur and a negative one, the case in which a polarity shift takes place.

3 Experiments and results

3.1 Evaluation corpus

Although the MPQA [15] corpus has been used in fine-grained opinion mining experiments, such as those presented in [8] and [13], most of them are directed to opinion holder and opinion expression identification and the targets identified in the MPQA corpus are less structured and can vary from named entities to abstract concepts described in a larger text span. For these reasons, we chose the JDPA [6] corpus as an evaluation benchmark for our classification experiments. The creators of the corpus provide details about it in [6].

From the JDPA corpus, we extracted the sentiment expression and their targets. To respect our proposed workflow described in the previous section, we replaced the actual sentiment expressions and targets with abstract tokens, "SENTIMENT" and "TARGET", respectively. Due to the high number of annotated sentiments and entities, we used for our test the "camera" set of files from the JDPA corpus. For the polarity shift identification task,

the extraction of the positive and negative instances is done by using the negation indicators from the JDPA corpus and replacing any sentiment expression with the SENTIMENT token. We replaced the negation identifier with the NEGATION token, whereas for the relation identification task we replaced the target expression with the TARGET token.

3.2 Relation classification results

In Table 1, we provide an overview of the best result for each method that we described in the previous section. For the tree kernel experiments, T represents the parse tree, V1 represents a one dimensional feature vector consisting of the number of tokens between the SENTIMENT and the TARGET tokens and V2 a two dimensional vector that also contains the number of punctuation marks between the SENTIMENT and the TARGET tokens. As it can be observed, the SVM with the tree kernel together with the two distance features provide the best results for the accuracy, precision and recall.

Table 1 Overview of the best result for each method.

Base Model	Variation	Accuracy	Precision	Recall
Classic Bag-of-Words	Naïve Bayes	79.82	80.2	79.8
	SVM + RBF	76.2	79.8	76.2
	SVM + Poly.	78.4	79.6	78.4
Proximity Bag-of-Words	Naïve Bayes	82.5	84.2	82.6
	SVM + RBF	83.09	83.5	83.1
	SVM + Poly.	78.27	81.6	78.3
SVM + Tree Kernel	T	83.896	83.684	85.488
	T + V1	86.182	86.034	87.534
	T + V2	**86.442**	**86.332**	**87.708**

In our experiments, we used the occurrence bag-of-words model because we dealt with small sentences and the other two types brought little new information for the classifier. For the feature dictionary generation, we used the lemma of the words that appeared in all of the sentences.

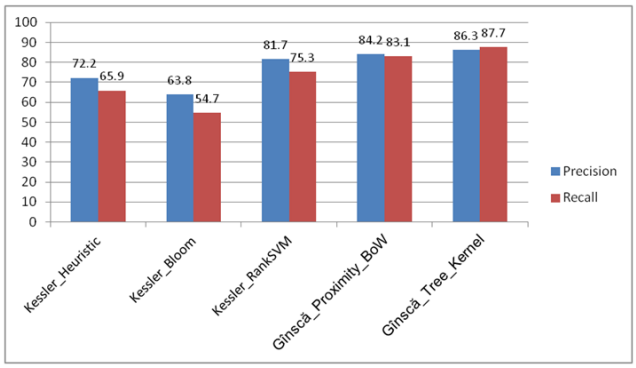


Figure 2 Comparison with Kessler's top 3 results.

We compare our best results to those reported by the authors of the JDPA corpus in their 2009 paper [7]. We retained the results from the best 3 methods that they have used: Heuristic, Bloom and Rank SVM.

The results presented in Figure 2 show that our two novel approaches to sentiment target identification, the proximity bag-of-words model and a tree kernel together with a feature vector composed of 2 elements outperform the top 3 approaches presented in [7].

3.3 Opinion polarity shift identification results

Given the fact that the tree kernel experiments provided the best results, we used for the polarity shift identification problem the same classification configurations as in section 4.3, for the relation identification task. In Table 2, we show the accuracy, precision and recall results for the tree kernel polarity shift identification experiments.

Table 2 10 Fold cross validation results for tree kernel polarity shift identification.

Window Size	Features	Accuracy	Precision	Recall
2	T	84.39	85.94	84.5
	T + V1	87.28	88.45	86.68
	T + V2	87.64	87.25	86.2
1	T	85.67	85.40	85.62
	T + V1	**89.8**	**90.25**	**88.72**
	T + V2	89.4	89.92	88.25
0	T	85.05	87.32	86.48
	T + V1	86.48	87.48	86.95
	T + V2	86.5	87.05	87.25

Because the negation identifier is regularly closer to the sentiment expression than the target is to the sentiment expression and the words before the negation and those after the sentiment expression have less influence on these, we chose to test a window size of maximum 2. The windows size represents the number of tokens before the first appearance and the number of tokens after the last appearance of the SENTIMENT or the TARGET tokens that are taken into consideration for classification from the whole sentence.

4 Conclusion and future work

We described in this paper a novel approach to fine-grained opinion mining that, after an initial step that involves the use of lexical resources, treats the problem of finding the opinion expressed towards an entity as a relation classification task. We detailed our classification workflow, a novel proximity bag-of-words model and we presented how tree kernels can be successfully used for relation classification, as well as for polarity shift identification. We included an overview of the best result obtained when using each method and we showed that both of our two novel approaches to the detection of sentiments expressed towards a certain target outperformed the methods proposed by the authors of the evaluation corpus.

Due to the fact that the best results were obtained when we used a tree kernel together with feature vectors, we plan to investigate the impact of using other features than those presented in this paper. So far, we focused our research on sentiment target identification but the same methods we used for this task can be used for another aspect of fine-grained opinion mining, opinion holder identification. This is a direction worth pursuing.

References

1 J. Bollen, A. Pepe, and H. Mao. Modeling public mood and emotion:twitter sentiment and socioeconomic phenomena. *Proceeding of the WWW Conference*, 2010.

2 A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, 2004.

3 S. Gerani, M. Carman, and F. Crestani. Proximity-based opinion retrieval. *Proceeding of the Special Interest Group on Information Retrieval Conference (SIGIR)*, 2010.

4 X. Huang and W. Croft. A unified relevance model for opinion retrieval. *Proceedings of the ACM Conference on Information and Knowledge Management*, 2009.

5 T. Joachims. Making large-scale svm learning practical. *Advances in Kernel Methods - Support Vector Learning*, 1999.

6 J. Kessler, M. Jason, L. Clark, and N. Nicolov. The 2010 icwsm jdpa sentment corpus for the automotive domain. *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media Data Workshop Challenge (ICWSM-DWC 2010)*, 2010.

7 J. Kessler and N. Nicolov. Targeting sentiment expressions through supervised ranking of linguistic configurations. *Proceedings of the 3rd Int'l AAAI Conference on Weblogs and Social Media*, 2009.

8 S.-M. Kim and E. Hovy. Automatic detection of opinion bearing words and sentences. *Companion Volume to the Proceedings of IJCNLP-05, the Second International Joint Conference on Natural Language Processing*, pages 61–66, 2005.

9 A. Moschitti. A study on convolution kernels for shallow semantic parsing. *Proceedings of the 42-th Conference on Association for Computational Linguistic (ACL-2004)*, 2004.

10 A. Moschitti. Making tree kernels practical for natural language learning. *EACL 2006, 11st Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference*, 2006.

11 S. O. Orimaye. Sentence-level contextual opinion retrieval. *Proceedings of the 20th international conference companion on World Wide Web*, 2011.

12 B. Pang and L. Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, 2008.

13 V. Stoyanov and C. Cardie. Topic identification for fine-grained opinion analysis. *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 817–824, 2008.

14 M. Tsytsarau and T. Palpanas. Survey on mining subjective data on theweb. *Data Mining and Knowledge Discovery Special Issue*, 2011.

15 J. Wiebe and C. Cardie. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2-3):165–210, 2005.

16 D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, pages 1083–1106, 2003.

Mechanisms for Opponent Modelling

Christos Hadjinikolis[1], Sanjay Modgil[1], Elizabeth Black[1], and Peter McBurney[1]

1 **Department of Informatics, King's College London, UK**
`name.surname@kcl.ac.uk`

Abstract

In various competitive game contexts, gathering information about one's opponent and relying on it for planning a strategy has been the dominant approach for numerous researchers who deal with what in game theoretic terms is known as the *best response problem*. This approach is known as *opponent modelling*. The general idea is given a model of one's adversary to rely on it for simulating the possible ways based on which a game may evolve, so as to then choose out of a number of response options the most suitable in relation to one's goals. Similarly, many approaches concerned with strategising in the context of dialogue games rely on such models for implementing and employing strategies. In most cases though, the methodologies and the formal procedures based on which an opponent model may be built and updated receive little attention, as they are usually left implicit. In this paper we assume a general framework for argumentation-based persuasion dialogue, and we rely on a logical conception of arguments—based on the recent *ASPIC*$^+$ model for argumentation—to formally define a number of mechanisms based on which an opponent model may be built, updated, and augmented.

1998 ACM Subject Classification I.2. Computing Methodologies Artificial Intelligence

Keywords and phrases dialogue, strategies, argumentation, opponent model

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.62

1 Introduction & Related work

Numerous researchers who deal with the best response problem rely on opponent modelling for implementing, employing and analysing strategies [1, 4, 3, 8, 10, 11, 12]. Essentially, an opponent model (OM) consists of four basic components: an opponent's knowledge; abilities; objectives, and; strategy. However, in most cases the methodologies and the formal procedures based on which such a model may be built and updated are often either left implicit, or are just concerned with particular components of the model.

Specifically, in the context of argumentation-based dialogue games, Riveret *et al.* [10, 11] model the possible knowledge of their opponents in the form of arguments, assuming that arguers are perfectly informed about all the arguments previously advanced by all other players. Their investigation concerns *games of perfect information*, and assumes that the participants' goals always comply with the dialogical objectives of the game, an assumption which, as McBurney *et al.* argue in [7], does not always hold. Oren *et al.* [8] present a generally complete approach through modelling both an agent's knowledge in the form of arguments as well as their goals, while in a similar sense to [4] they also allow for nested OMs. Additionally, they argue that given the knowledge about an opponent's goals it is also possible to indirectly model its strategy. However, nowhere in the aforementioned work is the problem of acquiring and maintaining an OM discussed. An interesting exception proposed by Black *et al.* [1] concerns a mechanism that enables agents to model *preference* information about others—what is important to another agent—and then rely on this information for making

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 62–68

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

proposals that are more likely to be agreeable. In their case the mechanism responsible for modelling an agent's preferences is explicitly provided.

In this work, we attempt to formally address the problem of opponent modelling through providing two mechanisms concerned with how an OM may be built, updated and possibly augmented. The rest of the paper is structured as follows. In Section 2 we briefly present the basic components of an $ASPIC^+$-based framework for persuasion dialogue. In Section 3 we present two mechanisms responsible for building, updating and augmenting an OM. Finally, in Section 4 we summarise and discuss future directions for our work.

2 An *ASPIC*$^+$ framework for persuasion dialogues

In [9] Prakken describes an argumentation framework (AF) by assuming an unspecified *logical language* ($\mathcal{L}$) and by defining arguments as inference trees formed by applying *strict* ($\mathcal{R}_s$) or *defeasible* ($\mathcal{R}_d$) inference rules of the form $\varphi_1, \ldots, \varphi_n \rightarrow \varphi$ respectively $\varphi_1, \ldots, \varphi_n \Rightarrow \varphi$. To define attacks between arguments, minimal assumptions on $\mathcal{L}$ are made; namely that certain well formed formulæ (wff) are a contrary or contradictory of certain other wff. Apart from this the framework is still abstract: it applies to any set of $\mathcal{R}_s$ and $\mathcal{R}_d$ and to any $\mathcal{L}$ with a defined contrary relation. Arguments are then constructed with respect to a knowledge base that is assumed to contain three kinds of premises that are wff in $\mathcal{L}$. These premises are expressed through a set of three disjoint subsets $\mathcal{K} = \mathcal{K}_n \cup \mathcal{K}_p \cup \mathcal{K}_a$: $\mathcal{K}_n$ is the (necessary) *axioms* (which cannot be attacked); $\mathcal{K}_p$ is the *ordinary premises* (on which attacks succeed contingent upon satisfying certain preference criteria[1]), and; $\mathcal{K}_a$ is the *assumptions* (on which attacks are always successful, *cf.* assumptions in [2]). Thus an $ASPIC^+$ argument in a set $\mathcal{A}$ of arguments that may be constructed from the aforementioned logical components, may either be a single premise, or a chain of premises and rules that lead to a certain conclusion. Lastly, three kinds of *attack* are defined for arguments. $B \in \mathcal{A}$ can attack $A \in \mathcal{A}$ by attacking a premise or conclusion of A, or a defeasible inference step in A. Some kinds of attack succeed as *defeats* independently of preference criteria, whereas others succeed only if the attacked argument is not stronger than the attacking argument.

We assume a general framework for persuasion dialogue where the participants, a proponent (P) and an opponent (O), debate the truth of a claim φ through exchanging dialogue moves ($\mathcal{DM}$s) consisting of arguments, based on the attack relationship between them and on a set of protocol rules that regulate the dialogue process; i.e. a participant can introduce an argument into the game if it attacks another argument that was previously introduced into the game by its interlocutor. We assume that the participants share the same $\mathcal{L}$ and the same contrary relation definition; i.e. there is agreement as to whether a given argument attacks another. In this respect, we define a dialogue $\mathcal{D}$ as a sequence of dialogue moves $< \mathcal{DM}_0, \ldots, \mathcal{DM}_n >$, where the content of $\mathcal{DM}_0$ is an argument for φ, while we assume that the dialogue process is regulated by a multi-reply protocol. The latter means that *backtracking* is allowed, which implies that a participant can return to a previous point in the game and attack against a previous move of its interlocutor in a different way. Thus a dialogue may also be expressed in the form of a *dialogue tree* ($\mathcal{T}$), as the one illustrated in Figure 1(a), where each node is a $\mathcal{DM}$ and each arc indicates a move's target (a backtracking example is P's argument G introduced by $\mathcal{DM}_6$ in Figure 1(a)). Full details of the $ASPIC^+$ framework as well as of the proposed dialogue framework can be found in [9] and [5] respectively.

[1] An important feature of the $ASPIC^+$ framework is the employment of preference-orderings over defeasible rules and non-axiom premises which we do not take into account for the purpose of this paper.

We assume that the accumulated logical information introduced by a participant in $\mathcal{D}$ is stored in a *commitment store* which we define as follows:

▶ **Definition 1.** Given a set of agents $Ags = \{Ag_1, \ldots, Ag_\nu\}$ participating in $\mathcal{D} = < \mathcal{DM}_0, \ldots, \mathcal{DM}_n >$, then for any agent Ag_i we define its **commitment store** as a tuple $CSi^t =< \mathcal{K}i^t, \mathcal{R}i^t >$, where $\mathcal{K}i^t$, and $\mathcal{R}i^t$, are respectively the premises and the rules moved into the game by Ag_i up to turn t, for $t = 0 \ldots n$, such that $CSi^0 = \emptyset$, and CSi^{t+1} is obtained by augmenting CSi^t with the logical information provided by the dialogue move $\mathcal{DM}_{t+1}$.

Finally, we assume that each agent $Ag_i \in Ags$ can engage in dialogues in which its strategic selection of moves may be based on what Ag_i believes its interlocutor (in the set $Ag_{j \neq i}$) knows. Accordingly, and in a similar sense to the approach employed in [8], each Ag_i maintains a model of its possible opponent agents. In contrast with [8], the proposed model consists of the goals and knowledge other agents may use to construct arguments, rather than just the abstract arguments and their relations.

▶ **Definition 2.** Let $Ags = \{Ag_1, \ldots, Ag_\nu\}$ be a set of agents. For $i = 1 \ldots \nu$, the **knowledge base** $\mathcal{KB}$ of Ag_i is a tuple $\mathcal{KB}_i =< S_{(i,1)}, \ldots, S_{(i,\nu)} >$ such that for $j = 1 \ldots \nu$, each sub-base $S_{(i,j)} = \langle \mathcal{K}_{(i,j)}, \mathcal{R}_{(i,j)}, \mathcal{G}_{(i,j)} \rangle$ is an OM expressing what Ag_i believes is Ag_j's premises ($\mathcal{K}_{(i,j)}$), rules ($\mathcal{R}_{(i,j)}$), and goals ($\mathcal{G}_{(i,j)}$), and where $S_{(i,i)}$ represents Ag_i's own beliefs and goals.

3 Modelling mechanisms

We begin by associating a *confidence* value c to the logical components of the information sets found in a sub-base $S_{(i,j)}$. Essentially, for an agent Ag_i this value expresses the probability of a certain logical component in $S_{(i,j)}$ being part of Ag_j's actual knowledge. For the computation of this value we differentiate between whether a particular information is: gathered directly by Ag_i, on the basis of its opponent's updated commitment store, or; a result of an *augmentation* attempt of Ag_i's current model of Ag_j. The latter concerns an incrementation of a current OM with the addition of arguments that are *likely* to also be known to Ag_i's opponent.

Intuitively, in real life we tend to assume that certain information, if known, is then likely to be related with other information, i.e. that there is some *relevance* between distinct pieces of information, which in our case may be translated as relevance between arguments. For example, assume that two agents, Ag_i (a proponent) and Ag_j (an opponent), engage in a dialogue as the one described in Figure 1(a), where Ag_i and Ag_j introduce arguments $\{A, C, E, G\}$ respectively $\{B, D, F, H\}$. Assume then, that Ag_i engages in another dialogue with the same root move A, but with a different agent Ag_m who also happens to counter A with argument B. It is then reasonable to assume that Ag_m is *likely* to also know of arguments D, H or even F. In this respect, the basic idea is, given an OM—which in essence describes a set of arguments known to one's opponent—and a mapping of a broader set of arguments with respect to a relevance factor, to then augment the OM by including arguments—and thus the logical elements that compose them—that have a high probability to also be known to that opponent, based on their relevance relationship with arguments already in the OM.

For assigning a confidence value c to the elements of an $S_{(i,j)}$, we will assume that every agent *retains* its own rules and premises without revision but relies on argumentation theory and semantics for resolving conflicts. In this respect, an agent's beliefs are formed based on deciding on the acceptability level of its arguments according to a number of different acceptability semantics. Thus in the face of new information *nothing is replaced or discarded*,

but instead certain arguments may simply cease to be or may become acceptable under different semantics. We will therefore assume that the confidence value of information acquired directly from the commitment store of one's interlocutor is equal to 1, which represents the highest level of confidence. However, this assumption must exclude information concerned with goals ($\mathcal{G}$) as those cannot be retained in the face of conflicts, i.e. it is not reasonable for an agent to be in pursue of conflicting goals at the same time. We leave the provision of a function for updating an opponent's goals to future work.

▶ **Definition 3.** Assume an $S_{(i,j)} \in \mathcal{KB}_i$, then for $Y \in \{\mathcal{K}_{(i,j)}, \mathcal{R}_{(i,j)}, \mathcal{G}_{(i,j)}\}$, X is a tuple $< x, c >$ where $x \in Y$, and where c represents the **confidence level** of x such that:

$$c^{[0,1]} = \begin{cases} 1 & \text{if } x \text{ is directly collected by } Ag_i \quad (a) \\ Pr(x) & \text{if } x \text{ is part of an augmentation of } S_{(i,j)} \quad (b) \end{cases}$$

where $Pr(x)$ is the likelihood of x being also known to Ag_j, given its current OM ($S_{(i,j)}$).

▶ **Definition 4.** Let Ag_i and Ag_j be two agents in Ags such that $1 \leq i, j \leq \nu$ and $i \neq j$, and $h_{(i,j)} =< \mathcal{D}^1, \ldots, \mathcal{D}^{\mu-1} >$ be Ag_i's history of dialogues with Ag_j. Then, given the current version of sub-base $S^{\mu-1}_{(i,j)} = \langle \mathcal{K}^{\mu-1}_{(i,j)}, \mathcal{R}^{\mu-1}_{(i,j)}, \mathcal{G}^{\mu-1}_{(i,j)} \rangle$ and the commitment store $CSj =< \mathcal{K}j, \mathcal{R}j >$ of the latest dialogue $\mathcal{D}^\mu$, Ag_i can **update** its sub-base $S^{\mu}_{(i,j)} = \langle \mathcal{K}^{\mu}_{(i,j)}, \mathcal{R}^{\mu}_{(i,j)}, \mathcal{G}^{\mu}_{(i,j)} \rangle$ such that: **(a)** $\mathcal{K}^{\mu}_{(i,j)} = \mathcal{K}^{\mu-1}_{(i,j)} \cup \mathcal{K}j$, and; **(b)** $\mathcal{R}^{\mu}_{(i,j)} = \mathcal{R}^{\mu-1}_{(i,j)} \cup \mathcal{R}j$.

For augmenting a current OM, we rely on a *relevance graph*.

▶ **Definition 5.** For an agent Ag_i, let $\mathcal{H}_i = \{h_{(i,1)}, \ldots, h_{(i,\nu)}\}$ be the set of all its histories, then an **abstract relevance graph (ARG)** is a weighted directed graph $\mathcal{G} = \{V, R\}$, where V is a set that consists of all arguments $A^{\mathcal{H}}$ encountered by Ag_i in $\mathcal{H}_i$, and where R is a set of weighted arcs, each of them indicating a relevance relationship between two arguments in $\mathcal{G}$, based on a weight function w, such that $w : R \to [0, 1]$.

3.1 Building a relevance graph

We assume an ARG to be incrementally built as an agent Ag_i engages in numerous dialogues, being empty at the beginning, and constantly updated with newly encountered opponent arguments (OAs). Notice that OAs appear only in the odd levels of a tree (Figure 1(a)). For assigning arcs between these arguments one may rely on how and when an argument appears in a dialogue tree. Specifically, we rely on the following condition:

▶ **Condition** 1. Given a dialogue tree $\mathcal{T}$, then for any argument A that appears in level i, and any argument B that appears in level j, for i and j being odd numbers and $j \geq i$, if $\frac{j-i}{2} \leq n$, and there exists a path between A and B in $\mathcal{T}$, there is an arc from A to B in $\mathcal{G}$.

Figures 1(b) and 1(c) illustrate two distinct ARGs induced from the dialogue tree of Figure 1(a), for $n = 1$ and $n = 2$ respectively. Intuitively, this modelling approach simply reflects the implied relationship that consecutive OAs have in a single branch of a tree. Through modifying the n value one can strengthen or weaken the connectivity, and so the relationship, between arguments in the induced ARG. However, one may choose to deviate from this particular modelling approach, adopting a different one so as to reflect a different kind of implied relationship between arguments. Lastly, for a pair of arguments $\{A, B\}$ connected with an arc r, let $w(r_{AB})$ be the weight value of an arc r which extends from argument A to argument B, we assume $w(r_{AB})$ to be equal to the number of agents N_{AB} that have moved A followed by B in a dialogue game, thus satisfying the relevance condition for $n = 1$, against the total number of agents $|Ags|$ minus agent Ag_i, i.e.

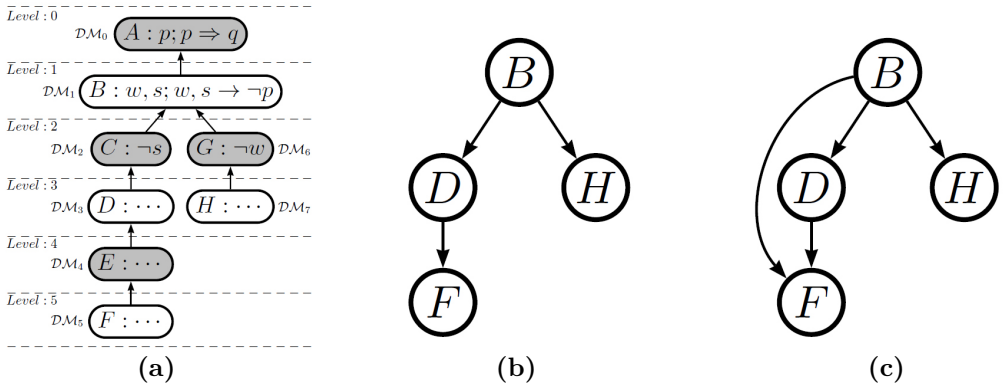


Figure 1 **(a)** A dialogue tree $\mathcal{T}$ where the grey and the white nodes concern P's respectively O's moves, **(b)** A 1-hop *ARG* modelling approach **(c)** A 2-hop *ARG* modelling approach.

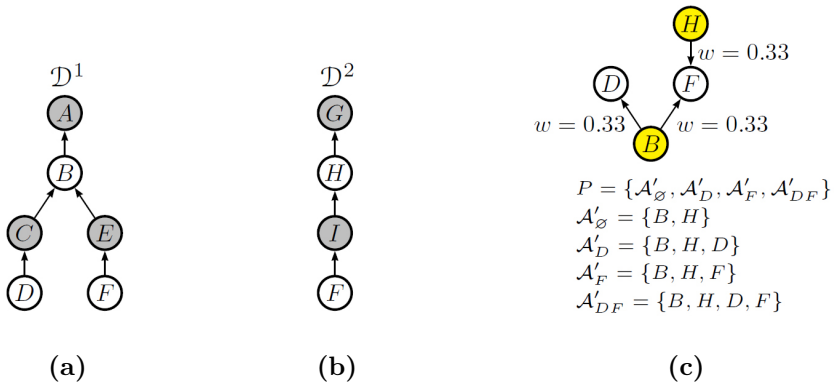
$w(r_{AB}) = N_{AB}/(|Ags| - 1)$. Dividing N_{AB} with $(|Ags| - 1)$ is necessary for normalising the arcs' weight values into probabilities.

3.2 Relevance augmentation

Given an ARG an agent Ag_i can then attempt to augment its OM of Ag_j (i.e. its $S_{(i,j)}$) by adding to it the logical information comprised in the arguments (nodes) that are of 1-hop distance in $\mathcal{G}$ from those that can be constructed from $S_{(i,j)}$. In a trivial case, let $Ags = \{Ag_1, Ag_2, Ag_3, Ag_4\}$, and $\mathcal{G}$ be an ARG induced by Ag_1 based on dialogues $\mathcal{D}^1$ and $\mathcal{D}^2$ for $n = 1$, as it appears in Figure 2(c). Let $S_{(1,4)}$ be Ag_1's OM of Ag_4 such that Ag_1 believes that Ag_4 can construct two arguments $\mathcal{A} = \{B, H\}$. Thus Ag_1's OM of Ag_4 can be expressed as a sub-graph $G_{\mathcal{A}} = \{\mathcal{A}, \varnothing\}$ of $\mathcal{G}$ (the yellow nodes in Figure 2(c)). Hence, Ag_1 computes the *likelihood* of each of the possible augmentations $\mathcal{A}' \in P$ of $\mathcal{A}$ as those appear in set $P = \{\mathcal{A}'_{\varnothing}, \mathcal{A}'_D, \mathcal{A}'_F, \mathcal{A}'_{DF}\}$ (Figure 2(c)), and selects the one with the highest likelihood for augmenting $S_{(1,4)}$. Given that the instantiation of an augmented OM relies on the arcs' weights of $\mathcal{G}$, we have to provide an arc-centric formula for computing this probability, as there are multiple ways based on which a particular augmentation may be induced. In other words and in graph theoretic terms, a possible augmentation of an OM is interpreted as a possible *graph expansion*. Thus a certain $\mathcal{A}'$ may be induced as a result of numerous possible expansions of $G_{\mathcal{A}}$, each containing *different arcs* while having the same set of arguments.

For example, assume we want to calculate the likelihood of augmentation $\mathcal{A}'_F = \{B, H, F\}$. Let $S(\mathcal{A}'_F) = \{G_1, G_2, G_3\}$ be the possible expansions of $G_{\mathcal{A}}$ which induce $\mathcal{A}'_F$ by including in $G_{\mathcal{A}}$: either only r_{BF} creating G_1; either only r_{HF} creating G_2, or; only r_{BF} and r_{HF} creating G_3. Hence, the likelihood of $\mathcal{A}'_F$ is: $Pr(\mathcal{A}'_F) = Pr(G_1) + Pr(G_2) + Pr(G_3)$ $\Leftrightarrow Pr(\mathcal{A}'_F) = w(r_{BF}) \cdot (1 - w(r_{BD})) \cdot (1 - w(r_{HF})) + w(r_{HF}) \cdot (1 - w(r_{BF})) \cdot (1 - w(r_{BD})) +$ $w(r_{BF}) \cdot w(r_{HF}) \cdot (1 - w(r_{BD})) \Leftrightarrow Pr(\mathcal{A}'_F) = 0.35937$. Finally, the confidence value c of the newly included information in $S_{(1,4)}$ is assigned a value equal to the likelihood of the chosen augmentation as defined by Definition 3(b), i.e. $Pr(x) = Pr(\mathcal{A}')$.

For providing the general formula for computing the likelihood of a possible augmentation we rely on basic graph theory notation with respect to a node A in a graph $\mathcal{G}$, such as degree $d(A)$, adjacent vertices $N(A)$ where $|N(A)| = d(A)$, adjacent arcs $R(A)$, and arc weights $w(r)$. We additionally define N_S for a set of arguments S such that $N_S = \bigcup_{A \in S} N(A) | \{X \in N(A) : X \notin S\}$, and $R_S = \bigcup_{A \in S} R(A) | \{r_{AB} \in R(A) : B \notin S\}$. Additionally, let $\mathcal{A}$ be the



(a) **(b)** **(c)**

Figure 2 **(a)** A dialogue $\mathcal{D}^1$ between Ag_1 & Ag_2 **(b)** A dialogue $\mathcal{D}^2$ between Ag_1 & Ag_3 **(c)** An ARG induced by Ag_1 from $\mathcal{D}^1$ & $\mathcal{D}^2$, and the image of Ag_1's OM of Ag_4 on it (the yellow nodes B & H).

set of all arguments that may be induced from a single sub-base $S_{(i,j)}$, then given that an ARG is essentially built from a number of OMs, then it must hold that $\mathcal{A} \subseteq \mathcal{A}^{\mathcal{H}}$. Provided $\mathcal{A}$, we assume $\mathcal{A}'$ to be an augmentation of $\mathcal{A}$ based on $\mathcal{G}$, such that $\mathcal{A}' = \mathcal{A} \cup S$ for $S \subseteq N_{\mathcal{A}}$. In this sense, we assume $G_{\mathcal{A}} = \{\mathcal{A}, \varnothing\}$ to be a sub-graph of $\mathcal{G}$ representing an image of an agent's Ag_i current OM of another agent Ag_j in $\mathcal{G}$, while we also assume $G_{\mathcal{A}'} = \{\mathcal{A}', R_i\}$ to be a possible expansion of $G_{\mathcal{A}}$, where $R_i \subseteq R_{\mathcal{A}}$. Given these, let $P = \{\mathcal{A}'_0, \mathcal{A}'_1, \ldots, \mathcal{A}'_{\mu-1}\}$ be the set of all possible distinct augmentations of $\mathcal{A}$, then the number of all possible distinct expansions of $G_{\mathcal{A}}$ with respect to neighbouring nodes that are of 1-hope distance from it, is:

$$\mu = |P| = \sum_{i=0}^{|N_{\mathcal{A}}|} \binom{|N_{\mathcal{A}}|}{i} \tag{1}$$

Furthermore, let $S(\mathcal{A}') = \{G^1_{\mathcal{A}'}, \ldots, G^n_{\mathcal{A}'}\}$ be a set of graphs containing all expanded graphs that have the same set of arguments $\mathcal{A}'$ such that $G_{\mathcal{A}'} = \{\mathcal{A}', R_j\}$ for $R_j \subseteq R_{\mathcal{A}}$, then the general formula for computing the likelihood of a possible augmentation is:

$$Pr(\mathcal{A}') = \sum_{G^{i=1}_{\mathcal{A}'} \in S(\mathcal{A}')}^{n} \left(\prod_{r \in R_j} w(r) \cdot \prod_{r \in R_{\mathcal{A}}/R_j} (1 - w(r)) \right) \tag{2}$$

Finally, since the likelihood of each possible augmentation should define a distribution of likelihoods then it must hold that:

$$\sum_{\mathcal{A}'_{i=0}}^{\mu-1} Pr(\mathcal{A}'_i) = 1 \tag{3}$$

4 Conclusions & Future direction

In this work we have addressed the problem of building, updating and augmenting an OM in argumentation-based dialogues. We relied on a logical conception of arguments based on the recent $ASPIC^+$ model for argumentation, and provided two modelling mechanisms: an update mechanism, and; an augmentation mechanism.

We have particularly focused on the latter, which relies on the relevance between information and attempts an augmentation of a current OM through the addition of information.

The latter is based on computing the likelihoods of a set of possible augmentations and choosing the one with the highest value. A drawback of the proposed approach is that, given a $\mathcal{G}$, all possible augmentations of a set $\mathcal{A}$ is equal to $|P| = 2^{|N(\mathcal{A})|}$ (each adjacent argument is either in or out of the augmentation), where $N_{\mathcal{A}}$ is the adjacent arguments of $\mathcal{A}$ (its neighbours), which implies that the complexity of computing the likelihoods of all possible augmentations of $\mathcal{A}$ increases exponentially as the number of the 1-hop neighbours of $\mathcal{A}$ increases. This makes the approach practically intractable.

However, drawing inspiration from the work of Li *et al.* [6] we intend to rely on an approximate approach for computing these likelihoods based on a Monte-Carlo simulation. Therefore our immediate future direction is to formally describe the exact simulation process for the proposed augmentation method. Additionally, we also intend to evaluate our approach through experimenting with software agents that engage in dialogue disputes. Particularly, we will compare the success rate of agents that rely on OMs and a relevance augmentation mechanism to agents who rely on simple opponent modelling and to agents who do not rely on opponent modelling at all.

References

1 E. Black and K. Atkinson. Choosing persuasive arguments for action. In *10th International Conference on Autonomous Agents and Multi-Agent Systems*, 2011.

2 Andrei Bondarenko, Phan Minh Dung, Robert Kowalski, and Francesca Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93(1–2):63–101, 1997.

3 D. Carmel and S. Markovitch. Model-based learning of interaction strategies in multi-agent systems. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(3):309–332, 1998.

4 David Carmel and Shaul Markovitch. Incorporating opponent models into adversary search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 120–125, Portland, Oregon, 1996.

5 C. Hadjinikolis, S.Modgil, E. Black, P. McBurney, and M. Luck. Investigating Strategic Considerations in Persuasion Dialogue Games. In *STAIRS 2012 - Proceedings of the Sixth Starting AI Researchers' Symposium*, volume 241 of *Frontiers in Artificial Intelligence and Applications*, pages 137–148. IOS Press, 2012.

6 Hengfei Li, Nir Oren, and Timothy J. Norman. Probabilistic argumentation frameworks. In *TAFA*, pages 1–16, 2011.

7 P. McBurney and S. Parsons. Games That Agents Play: A Formal Framework for Dialogues between Autonomous Agents. *Journal of Logic, Language and Information*, 11(3):315–334, 2002.

8 N. Oren and T. Norman. Arguing Using Opponent Models. In *Argumentation in Multi-Agent Systems*, volume 6057 of *Lecture Notes in Computer Science*, pages 160–174. 2010.

9 H. Prakken. An abstract framework for argumentation with structured arguments. *Argument and Computation*, 1(2):93–124, 2010.

10 R. Riveret, H. Prakken, A. Rotolo, and G. Sartor. Heuristics in argumentation: A game-theoretical investigation. In *Proceedings of COMMA*, pages 324–335, 2008.

11 R. Riveret, A. Rotolo, G. Sartor, H. Prakken, and B. Roth. Success chances in argument games: a probabilistic approach to legal disputes. In *Proceedings of the 20th annual conference on Legal Knowledge and Information Systems: JURIX*, pages 99–108, 2007.

12 William E. Walsh, Rajarshi Das, Gerald Tesauro, and Jeffrey O. Kephart. Analyzing complex strategic interactions in multi-agent systems. In *AAAI-02 Workshop on Game-Theoretic and Decision-Theoretic Agents*, Edmonton, 2002.

4D Cardiac Volume Reconstruction from Free-Breathing 2D Real-Time Image Acquisitions using Iterative Motion Correction

Martin Jantsch[1], Daniel Rueckert[1], and Jo Hajnal[2]

1 **Visual Information Processing Group, Department of Computing, Imperial College London**
Queen's Gate 180 London, UK
`www3.imperial.ac.uk/computing/`

2 **Imaging Sciences Division, King's College London**
St Thomas' Hospital London, UK
`www.kcl.ac.uk/medicine/research/divisions/imaging`

Abstract

For diagnosis, treatment and study of various cardiac diseases directly affecting the functionality and morphology of the heart, physicians rely more and more on MR imaging techniques. MRI has good tissue contrast and can achieve high spatial and temporal resolutions. However it requires a relatively long time to obtain enough data to reconstruct useful images. Additionally, when imaging the heart, the occurring motions - breathing and heart beat - have to be taken into account. While the cardiac motion still has to be correctly seen to asses functionality, the respiratory motion has to be removed to avoid serious motion artefacts.

We present initial results for a reconstruction pipeline that takes multiple stacks of 2D slices, calculates the occurring deformations for both cardiac and respiratory motions and reconstructs a coherent 4D volume of the beating heart. The 2D slices are acquired during free-breathing over the whole respiratory cycle, using a fast real-time technique. For motion estimation two different transformation models were used. A cyclic 4D B-spline free-form deformation model for the cardiac motion and a 1D B-spline affine model for the respiratory motion. Both transformations and the common reference frame needed for the registration are optimized in an interleaved, iterative scheme.

1998 ACM Subject Classification I.4.5 Transform methods

Keywords and phrases MRI, Cardiac, Registration

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.69

1 Introduction

Physicians are relying more and more on non-invasive imaging techniques to assess the functionality and morphology of the heart. In present practice echocardiography is still the standard, due to higher availability, lower costs and shorter acquisition and analysis times. But with increasing technical advances, both in acquisition hardware and image reconstruction and processing algorithms, MR imaging is becoming the favoured modality. MRI takes advantage of the magnetic properties of tissue, explicitly the signal response of hydrogen, which can be found in abundance in the whole human body, as it is mainly made out of water. During examination, the patient is placed inside a strong magnetic field, which aligns the otherwise unoriented nuclear spins of the nuclei. Gradient coils then spatially encode the signal produced by a radiofrequency exitation pulse, allowing the system

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 69–74

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

to gradually fill the so called K-space [6], which is a representation of the image in the Fourier domain. This raw signal can be Fourier transformed into an image, showing for example one slice of the human body or even a whole volume, depending on the setting. But the system has to wait for the tissue to reach a steady state again before continuing with the acquisition of the next spatial position. This can make MR imaging rather slow and prone to inconsistencies in the data, due to motion between acquisition steps, either in between different parts of K-space (results in blurring artifacts) or between slices (results in an inconsistent image volume). Especially imaging the beating heart, also moving due to respiration, poses a challenging problem, both from the acquisitional and reconstructional point of view.

Common methods as described in the literature to deal with this problem can be divided into two categories. The straight forward method requires the subject to hold its breath for about 20s. This produces very good results, but is not always feasible, as patients with heart problems are often not able to hold their breath long enough. Other ways to deal with respiratory motion rely on so called gating techniques [2]. They use some kind of surrogate signal (e.g. chest bellows or 1D pencil beam monitoring the diaphragm motion) to divide the respiratory cycle into small segments and only use data acquired at specific times (usually end-expiration). The problem with this approach is that it assumes that the breathing pattern is always the same, which of course is not the case (chest vs. abdominal breathing), resulting in minor motion artifacts. And the data from all other time points is either thrown away or not acquired at all, resulting in a much prolonged scanning time. In both cases the cardiac motion is usually dealt with by using the ECG signal to divide the cardiac cycle into small, near motion-free intervals and gradually filling the K-space of those time frames over a couple of heart beats [1].

We propose to use fast real-time imaging techniques to acquire individually motion-free slices [7], covering the heart volume in dense spatial and temporal positions. Those slices will be corrected for respiratory motion, with respect to a chosen reference breathing position and combined afterwards to form a complete 4D cardiac volume. To be able to register all images, we also have to estimate the cardiac motion towards a reference time point in the cardiac cycle. We model the cardiac motion by cyclic 4D B-Splines and choose an affine model for the respiratory motion.

2 Methods

The acquisition train is as follows: The scanner acquires 2D slices of the heart at a fixed spatial position using real-time techniques. To avoid gaps in the volume, due to translational motion, the respective slice is scanned over multiple heart beats (approximately one breathing cycle). After scanning one spatial position, we move to the next adjacent one. This scheme is necessary, as real-time techniques take some time to build up (to reach steady state). Thus the first couple of images are of inferior image quality and essentially useless for reconstruction. For further accelerate the individual slice acquisitions, we apply multiple receiver coils (SENSE) [8] and half-Fourier reconstruction.

We want to find a 3-dimensional reference volume $I_{0,0}$ and a time-dependent transformation $T_{k_i,l}$ that warps the 2-dimensional observed real-time images $I_{k_i,l}$ to locally fit this reference image. In the setting of free-breathing, cardiac cine MRI, we differentiate between two occurring motions, which we assume to be independent.

The first is the approximately periodic, non-rigid deformation of the heart T_{Φ^c} due to its self-induced contraction and following relaxation. We use a multi-level 4D cubic B-Spline

model [5] defined on a mesh of control points, where Φ^c is the vector of deformations in (x, y, z)-direction for all control points:

$$FFD(x, y, z, t) = \sum_{o=0}^{3}\sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} B_o(\tau)B_l(u)B_m(v)B_n(w)\Phi^c_{i+l,j+m,k+n,h+o} \tag{1}$$

Note that actually Φ^c also contains a fourth component for the time, which is also modeled by B-splines. But since we assume the time points of the slices, denoted by the index $k_i \in [0, 1]$ with the corresponding cardiac cycle given by $i \in \{0, ..., N_k\}$ to be known (as taken from the ECG), an optimization for these parameters is not necessary. Since we continuously acquire one slice for every time point during one cardiac cycle and then jump to the next slice position, we have to assume that the cardiac movement of the heart is identical over all heart beats, aside from differences in their duration. To enforce periodicity we simply change the neighborhood definition for the B-Splines, effectively forcing the last to be equal to the first temporal control point.

The second transformation is the temporally smooth and approximately affine motion $T^l_{\Phi^r}$ induced by breathing. Φ^r is the vector of the 12 degrees of freedom, namely translation, rotation, scale and screw for all coordinates (x, y, z) respectively, and the index l denotes the real time points in the acquisition starting with $l = 0$ for the first slice. With the use of a navigator l can also be mapped into a smaller 1D or 2D space according to the momentary breathing state. Although the respiratory motion includes some small, local, free-form deformation, an affine model defined in a bounding box around the heart is a close enough approximation to start with [4]. We will try to incorporate a non-rigid motion model for the breathing motion at a later stage, using the periodicity of the cardiac motion to separate both simultaneously occurring motions.

To solve for the different degrees of freedom given by $I_{0,0}$, T_{Φ^c} and $T_{\Phi^r_l}$ we use an iterative scheme that minimizes the following cost function:

$$(I_{0,0}, \{\Phi^c\}, \{\Phi^r\}) = \underset{I_{0,0},\{\Phi^c\},\{\Phi^r\}}{\operatorname{argmin}} C(\{I_{k_i,l}\}, I_{0,0}, \{\Phi^c\}, \{\Phi^r\}) \tag{2}$$

where

$$\begin{aligned} &C(I_{k_i,l}, I_{0,0}, \{\Phi^c\}, \{\Phi^r\}) = \\ &C_{data}(\{I_{k_i,l}\}, I_{0,0}, \{\Phi^c\}, \{\Phi^r\}) + \lambda_0 C_{img}(I_{0,0}) + \lambda_1 C_{card}(\{\Phi^c\}) + \lambda_2 C_{resp}(\{\Phi^r\}) \end{aligned} \tag{3}$$

The first term is a similarity criterion between the transformed reference volume and the observed, 2D slices. We chose the sum of squared differences (SSD) measure, since we are dealing with monomodal data and we can assume that the image intensities stay constant during motion:

$$C_{data}(\{I_{k_i,l}\}, I_{0,0}, \{\Phi^c\}, \{\Phi^r\}) = \frac{1}{|\mathcal{L}||\Omega|}\sum_{\{l\}}\sum_{\mathbf{x}\in\Omega_l}(I_{k_i,l}(T_{\Phi^c}(T_{\Phi^r}(\mathbf{x}, l), k_i)) - I_{0,0}(\mathbf{x}))^2 \tag{4}$$

where Ω_l is the 2D domain of the current slice in the 3-dimensional volume Ω and $|\mathcal{L}|$ and $|\Omega|$ are the amounts of temporal and spatial voxel coordinates respectively.

The second term in (3) is an optional regularization term for the reference image, that can be used in a super-resolution framework. For example an image gradient magnitude operator in an l_2-norm would be appropriate if the number of acquired slices is small, thus resulting in an underdetermined super-resolution volume reconstruction problem. It penalizes the high-frequency components in the estimated image.

The last two cost terms regularize the two occurring motions. The periodic B-spline description of the cardiac motion directly enforces spatial and temporal smoothness. Whether it is necessary to apply some regularization, for example enforcing diffeomorphic transformations or an incompressability constraint of the myocardium has yet too be determined.
For the respiratory motion we introduce a temporal smoothness penalty that ensures a slow and smooth evolution of the motion parameters, starting from an identity transformation for $l = 0$:

$$C_{resp}(\{\Phi^r\}) = \sum_{\{i\}} d\Phi^r_{l_i,l_{i+1}} ||\Phi^c_l - \Phi^r_{l+1}||^2_2 \tag{5}$$

where $d\Phi^r_{l_i,l_{i+1}} = \frac{1}{l_{i+1}-l_i}$ is a temporal normalization and i the set of indices of all ls. This is necessary as the temporal offsets between the acquisition of two slices are not always uniform, because in real-time imaging changing the slice position costs some time for the excitation to reach steady-state again. So if the temporal distance is big, the confidence in the solution is small and $d\Phi^r_{l_i,l_{i+1}}$ reduces the weight of the corresponding term.
In the above cost terms λ_0, λ_1 and λ_2 are regularization parameters, weighting the relative contributions of the corresponding terms. The parameters are chosen experimentally.

The iterative scheme to solve the least square problem (2) starts with an initial estimate of $I_{0,0}$ and Φ^r and alternates between optimizing the 3 different sets of parameters:

Step 1: $(\{\hat{\Phi}^c_{(n+1)}\}, \{\hat{\Phi}^r_{(n+1)}\}) = \underset{\{\Phi^c\},\{\Phi^r\}}{\operatorname{argmin}} C(\{I_{k_i,r}\}, I^{(n)}_{0,0}, \{\Phi^c\}, \{\Phi^r\})$

Step 2: Calculate new $I^{(n+1)}_{0,0}$ using Scattered Data Interpolation

Step 3: Check for stop criterion and if not full-filled increment n and go to Step 1

where the stop criterion is $||C^{(n+1)} - C^{(n)}|| < \epsilon$ with an empirically chosen ϵ (usually $\epsilon = 0.0001$).

To be able to capture larger deformations, the data is first divided into spatially and with regard to the cardiac phase temporally continuous blocks/volumes. In subsequent iterations, these blocks are partitioned into ever smaller blocks. This makes the algorithm much more robust and we are also able to blur the images in the through-plane direction, which is necessary to calculate large deformations in that direction. Additionally the image resolution starts with a larger one and is gradually decreased to its original setting, while the images are at the same time blurred with a Gaussian in all possible directions.

3 Results

So far we evaluated the performance of the algorithm without respiratory motion. Two data sets were used. Both are cardiac and respiratory gated. The first one is a 3D cine of the heart with resolution 1.25x1.25x2mm (fig. 1) and the second was acquired with real-time techniques depicting the volume by 14 adjacent slices with a resolution of 1.25x1.25x8mm (fig. 2). Fig. 2 shows the heart in end-systole from different orthogonal views. Due to the large slice thickness in Fig. 1 we have chosen here only one view (short-axis) at different positions in the volume also in end-systole. The end-systole time point was chosen because it has the largest deformations with regard to the reference time point. Although both volumes have a lot of tissue moving in and out of the field of view, the proposed registration method manages to capture the deformations of important structures like the myocardium which will be important for estimating the respiratory motion. This can be seen when comparing the yellow isolines (taken from the ground truth image) with the edges of e.g. the

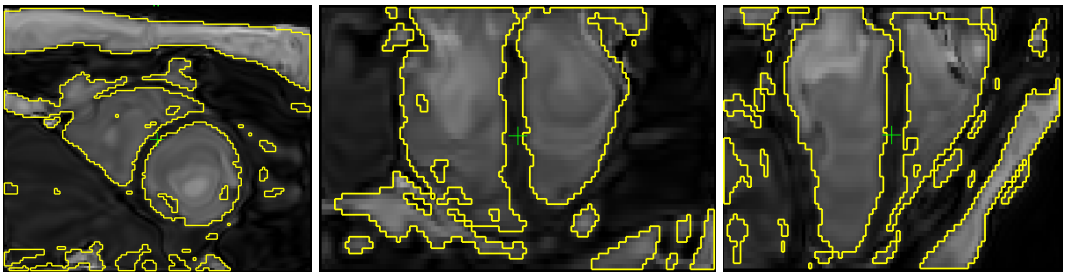


Figure 1 Registration result as tested on a 3D cine of the heart with resolution 1.25x1.25x2mm. From left to right: short-axis and 2 orthogonal long-axis views, all at end-systole. The yellow lines show the isolines of the ground truth.

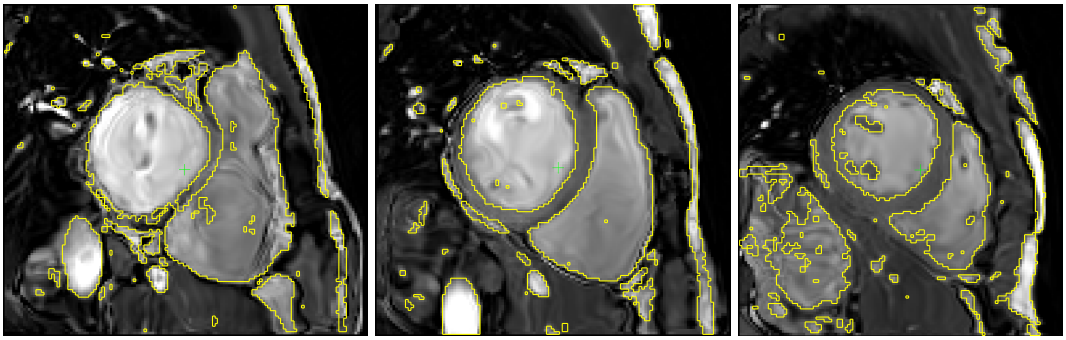


Figure 2 Registration result as tested on a stack of real-time images of the heart with resolution 1.25x1.25x8mm. Images were taken at different spatial positions of the volume, all at end-systole. The yellow lines show the isolines of the ground truth.

blood pool (lighter regions). When acquiring slices in the proposed way the sampling in the through-plane direction will be much denser, which will make it easier to accurately capture also the through-plane deformations.

4 Conclusion

The results show that the proposed method is able to accurately estimate the cardiac motion induced deformations. And it is robust towards through-plane motion and tissue moving in and out of the field of view, which is important for 2D to 3D registration. Jiang et al. [3] showed, for a similar problem the successful registration of 2D slices towards a successively updated reference frame using a linear transformation model. Based on those results, we are confident that we will be able to use the proposed iterative motion estimation scheme to estimate and correct for the respiratory motion.

Future work includes the successful computation of the affine transformation, tested with an example respiratory motion model applied on a common cardiac cine sequence and with real free-breathing data. And we want to incorporate super-resolution techniques into the framework to improve spatial and temporal resolution.

References

1 J. P. Earls, V. B. Ho, T. K. Foo, E. Castillo, and S. D. Flamm. Cardiac MRI: recent progress and continued challenges. *Journal of Magnetic Resonance Imaging*, 2002.

2 R. L. Ehman, M. T. McNamara, M. Pallack, H. Hricak, and C. B. Higgins. Magnetic resonance imaging with respiratory gating: Techniques and advantages. *American Roentgen Ray Society*, 1984.

3 S. Jiang, H. Xue, A. Glover, M. Rutherford, D. Ruecker, and J. Hajnal. MRI of moving subjects using multislice snapshot images with volume reconstruction (SVR): Application to fetal, neonatal, and adult brain studies. *IEEE Transactions on Medical Imaging*, 26(7), 2007.

4 K. McLeish, D. L. G. Hill, D. Atkinson, J. M. Blackall, and R. Razavi. A study of the motion and deformation of the heart due to respiration. *IEEE Transactions on Medical Imaging*, 2002.

5 D. Rueckert, L.I. Sonoda, C. Hayes, D.L.G. Hill, M.O. Leach, and D.J. Hawkes. Non-rigid registration using free-form deformations: application to breast MR images. *IEEE Transactions on Medical Imaging*, 1999.

6 A. Shankaranarayanan, O.P. Simonetti, G. Laub, J.S. Lewin, and J.L. Duerk. Segmented k-space and real-time cardiac cine MR imaging with radial trajectories. *Radiology*, 2001.

7 M. Uecker, S. Zhang, D. Voit, A. Karaus, K.-D. Merboldt, and J. Frahm. Real-time MRI at a resolution of 20 ms. *NMR Biomed.*, 2010.

8 Y. Wang. Description of parallel imaging in MR using multiple coils. *Magnetic Resonance in Medicine*, 2000.

Collecting battery data with Open Battery

Gareth L. Jones and Peter G. Harrison

Imperial College London, 180 Queen's Gate, London, SW7 2RH
{gljones,pgh}@doc.ic.ac.uk

Abstract

In this paper we present Open Battery, a tool for collecting data on mobile phone battery usage, describe the data we have collected so far and make some observations. We then introduce the *fluid queue* model which we hope may prove a useful tool in future work to describe mobile phone battery traces.

1998 ACM Subject Classification D.4.8 Queueing theory

Keywords and phrases battery model, battery data

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.75

1 Introduction and motivation

A recent Forrester study suggests that by 2016 a billion smartphones will be in use around the world [2]. Understanding battery behaviour and how devices are used (sometimes called *human battery interaction* [7]) is important to deliver improved performance in these devices.

Previous studies (e.g. Ferreira et al. [3]) have collected data under privacy agreements which do not allow the data to be shared outside the named researchers on the original proposal. This makes further work with the data hard. Data collected in our study is published under the PDDL on our website `http://www.openbattery.com/`, is in the public domain, and can be downloaded and redistributed freely.

In this paper we will make some observations about the data we have collected so far and then introduce the fluid queue modelling paradigm.

The authors of this paper previously published a result on how battery life of a device subject to random charging and discharging periods was affected by a power saving mode, implemented when power reserves fell below some threshold value [5]. In future work we intend to investigate fitting this model to our data.

2 Data collection

We have written an application for Android which logs battery usage data. The application listens for `ACTION_BATTERY_CHANGED` broadcasts and logs the battery state with timestamp each time the battery state changes. Data is saved locally and sent to our web server when the device is charging. A sample of the data collected is shown in Figure 1.

So far in this preliminary work, we have collected data for around 20 handsets for 3 months.

3 Observations on collected data

1. There is great variability in the number of data points logged. We observed a *Samsung GT-I9000* handset logging more than 1,000 data points a day (reporting regular small changes in voltage), while a *HTC Wildfire S A510e* logged nearer 100 data points a day.

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 75–80

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

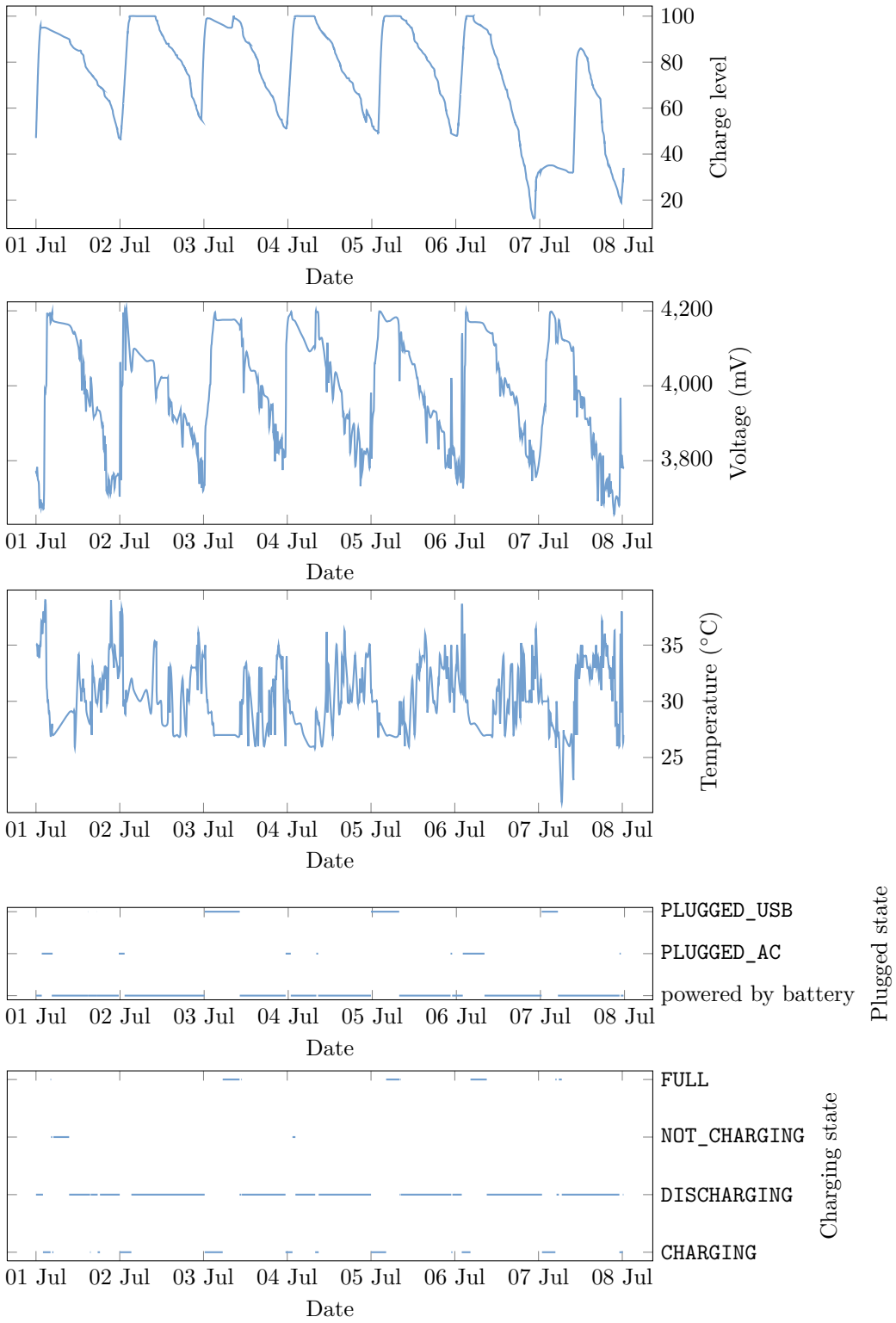


Figure 1 Sample data trace from July 1-7 2012 for device id `3fd6231afc7fec60`, a Galaxy Nexus. Throughout this trace the reported health was `GOOD`.

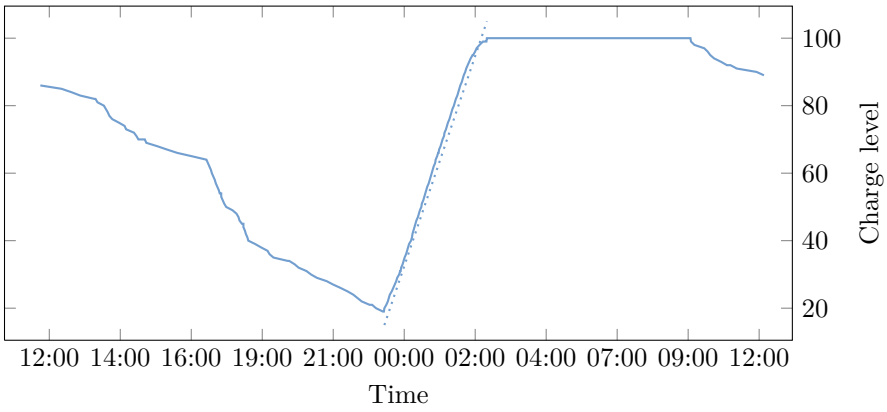


Figure 2 Linear charging/discharging period for Galaxy Nexus (device id `3fd6231afc7fec60`, 8–9 July 2012).

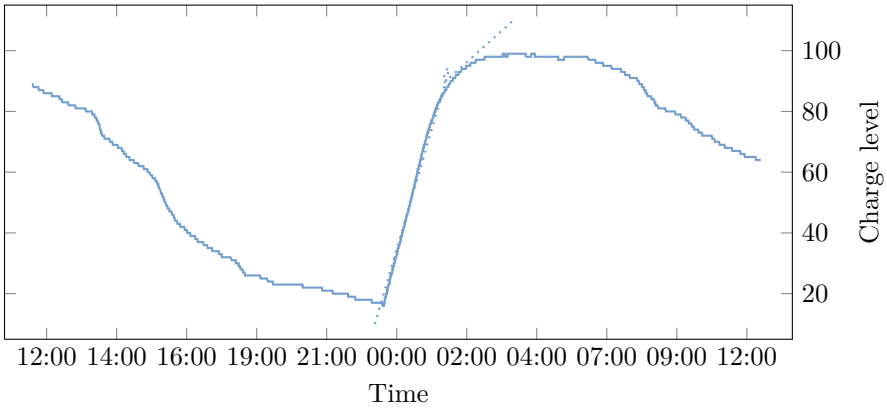


Figure 3 Non-linear charging for Nexus S (device id `3fe0f99cef3a49a8`, 11-12 July 2012) with two piecewise linear functions fitted.

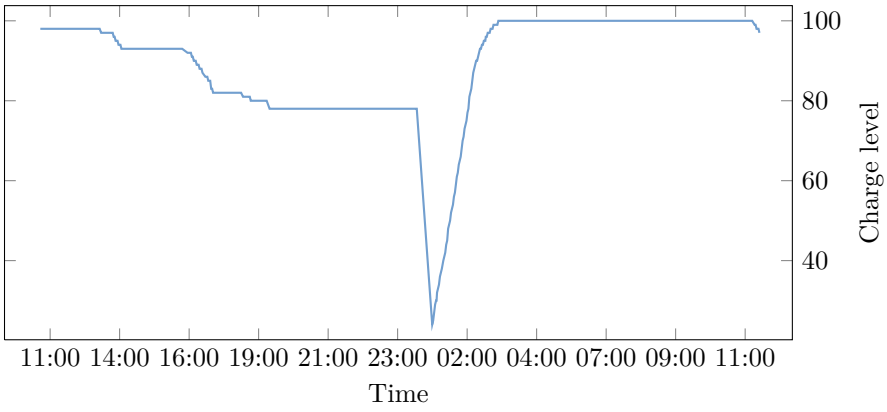


Figure 4 Erroneously reported discharging for *asus Transformer Prime TF201* (device id `3fcb2812d3d9e9b8`, 9-10 July 2012). From 19:00 to 23:20 the battery level was reported in the operating system as at a constant value.

HEALTH ∈ {UNKNOWN, GOOD, OVERHEAT, DEAD, OVER_VOLTAGE, UNSPECIFIED_FAILURE, COLD}.
PLUGGED ∈ {powered by battery, PLUGGED_AC, PLUGGED_USB}
STATUS ∈ {UNKNOWN, CHARGING, DISCHARGING, NOT_CHARGING, FULL}

Figure 5 `BatteryManager` health, plugged and status values.

Table 1 Average values computed for Galaxy Nexus.

State	Average change	Average filling/emptying time
AC charging	+30 pp/hour	3 hours 15 minutes to fully charge
USB charging	+20 pp/hour	5 hours to fully charge
discharging	−3 pp/hour	33 hours until fully drained

2. Generally reported charging rates are reasonably linear throughout the charging period such as for the Galaxy Nexus shown in Figure 2. However, data recorded for five Nexus S handsets shows linear charging up to 85%, with a non-linear portion up to 100%. As shown in Figure 3 we could reasonably approximate this with a second linear rate. Unsurprisingly we have less data for the lower end of battery charging (an earlier version of the logging application required manual restart), but are now aware that charging rates need to be level dependent.
3. There are also logging problems with data we need to consider. An *ASUS Transformer Prime TF201* (device id `3fcb2812d3d9e9b8`) has misreported the battery level as remaining constant for a few hours before dropping 50 percentage points or more in a matter of seconds as shown in Figure 4. This is not a bug with our tool, but with the levels that the battery hardware is reporting to the operating system.
4. Different handsets report their charging state differently. The values documented in the `BatteryManager` class are shown in Figure 5, but not all handsets report in the same way. For example, the *HTC Wildfire* (device id `3fe37029cced541a`) never reports itself `DISCHARGING`, only `CHARGING`, `NOT_CHARGING` or `FULL`.

4 Fluid queues

Fluid queues are a particular type of stochastic process which we hope will prove to be a good model for the charging and discharging behaviour seen in our data.

A fluid queue is a bivariate stochastic process (J_t, X_t) where J_t describes the background state and X_t the charge level. J_t is a Markov chain on the state space

{`AC charging`, `USB charging`, `discharging`}.

With each of these states we associate a rate of change which determines the rate at which X_t changes. In Table 1 we show parameters estimated from a 4 month trace from a Galaxy Nexus (device id `3fd6231afc7fec60`). Average charging rates for this device are broadly similar over all time periods, but average discharging rates varied significantly from 1 to 11 percentage points per hour (pp/hour).

A single exponential distribution holding time in each state is unlikely to describe the traces well, but extra states can be added within the fluid queue model to give a phase-type distribution fit to our data.

The process X_t is continuous and piecewise linear with the rate determined by the process J_t. The process is bounded above and below by the capacity of the battery ($0 \leq X_t \leq B$ for all t).

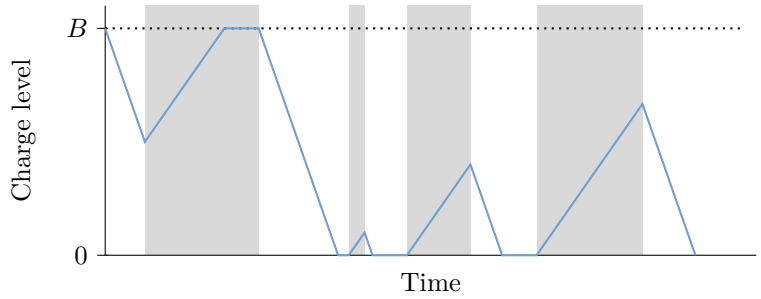


Figure 6 Sample trace from a fluid queue. The grey highlights represent time in charging periods and the white background periods when the device was discharging.

A sample trace from a fluid queue model with just two states {`charging`, `discharging`} is shown in Figure 6.

The *busy period* of the fluid queue is the time period between instants when the buffer is empty. The busy period is a stochastic quantity because it is determined by the sequence of charging and discharging period durations.

The fluid queue model has seen significant attention in the literature and the stationary distribution and busy period are known for infinite buffer models [4]. We will require an extension of the model introduced here where charging/discharging rates are dependent on the charge level and the buffer is of finite capacity.

Authors of this paper published the busy period for a model with level-dependent rates in a recent paper [5]. The Laplace–Stieltjes transform of the busy period distribution was computed, from which moments can be computed analyically by differentiation and numerical inversion can quickly compute particular values (e.g. 95th percentile). Extending this result to a finite buffer with numerous emptying states remains as future work.

5 Motivation and future goals

Smart phone user feedback on battery life is currently very crude. Android handsets typically warn the user of low battery at 15% and 5%, irrespective of how long the battery life is likely to last (on some handsets this might be 8 or more hours).

Some apps already exist to give users a clearer idea of how long their battery will last (like *Battery Monitor Widget* [1]), though they do not offer the user time-based alerts. A significanlty more elaborate system 'CABMAN' has been suggested [8] where the device would make decisions about power usage based on user position and proximity to the next charging session.

Our theoretical model may be can be improved by considering the 'phantom recharging' effects descibed by the KiBaM model [5, 6]. We see voltage recoveries during discharge periods in our collected data and intend to investigate the effect.

In the longer term we seek to investigate how the same device performs over time and between different users and quantitatively qualify the degradation in battery performance over time. This information will be of interest to both users and manufacturers of Android devices.

References

1 Battery monitor widget by 3c `http://www.3c71.com/android/?q=node/1`.
2 Forrester research mobile adoption forecast, 2012 to 2017 (US). 2012.

3 Denzil Ferreira, Anind K. Dey, and Vassilis Kostakos. Understanding human-smartphone concerns: a study of battery life. In *Pervasive'11 Proceedings of the 9th international conference on Pervasive computing*, pages 19–33, 2011.

4 Tony Field and Peter G. Harrison. Busy periods in fluid queues with multiple emptying input states. *Journal of Applied Probability*, 47:474–497, 2010.

5 Gareth L. Jones, Peter G. Harrison, Uli Harder, and A. J. Field. Fluid Queue Models of Battery Life. In *IEEE 19th International Symposium on Modelling, Analysis Simulation of Computer and Telecommunication Systems*, July 2011.

6 J. F. Manwell and J. G. McGowan. Lead acid battery storage model for hybrid energy systems. *Solar Energy*, 50(5):399–405, 1993.

7 Ahmad Rahmati, Angela Qian, and Lin Zhong. Understanding human-battery interaction on mobile phones. In *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, pages 265–272.

8 Nishkam Ravi, James Scott, Lu Han, and Liviu Iftode. Context-aware battery management for mobile phones. In *Sixth Annual IEEE International Conference on Pervasive Computing and Communications, 2008. PerCom 2008*, pages 224–233, 2008.

Informing Coalition Structure Generation in Multi-Agent Systems Through Emotion Modelling

Martyn Lloyd-Kelly and Luke Riley

Department of Computer Science
University of Liverpool, UK
mlk5060@liverpool.ac.uk and L.J.Riley@liverpool.ac.uk

Abstract

We propose a hybrid coalition formation method for multi-agent systems that combines a rational mechanism and an emotionally-inspired mechanism to reduce the associated computational cost. To initialise coalition formation, the rational mechanism is used and in subsequent iterations, the emotional mechanism (that forms coalitions resulting from emotional reactions to aspects of interactions between agents) is used. The emotions of *anger* and *gratitude* are modelled and used as a basis to model *trust* which is in turn used to restrict the coalition state-space. We offer some discussion as to how this hybrid method offers an improvement over using a method that only considers payoff maximisation and we propose some direction for future work.

1998 ACM Subject Classification I.2.11 Distributed Artificial Intelligence

Keywords and phrases Multi-Agent Systems, Coalition Formation, Emotion

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.81

1 Introduction

Multi-agent systems (*MAS*) are systems of autonomous agents capable of interacting with one another in some way [19]. In self-interested MAS, agents will attempt to achieve individual goals whilst maximising individual *payoffs*. However, under certain circumstances, agents may have to temporarily form mutually beneficial partnerships with other agents to achieve goals [3]. These partnerships are known as *coalitions* and a set of such coalitions are known as a *coalition structure*. Forming coalition structures in MAS has been shown to be an important topic that can be applied in many different areas. For example, [12] notes that it has proved useful in: e-commerce, e-business and distributed sensor networks.

In human societies, coalition formation takes into account rational aspects such as expected monetary payoffs as well as various *emotional dispositions* towards individuals. Emotions stemming from one individual's appraisal of another individual's actions, such as *gratitude* and *anger*, appear to be integral in the establishment and maintenance of *trust* [4] and such emotional appraisals, in conjunction with various cognition-based aspects, form the basis that enables one individual to trust another [6]. Trust formation of this kind has been tested using a theoretical framework by [9] who concludes that there is a high importance placed upon understanding the affective qualities of relationships. Therefore, whilst we can posit that coalitions in human societies are informed in part by maximisation of current finances, this does not fully encompass the whole spectrum of reasoning undertaken when forming coalitions as some consideration is also given to emotional aspects.

With regards to MAS, finding the best coalition structure i.e. the one that maximises social welfare, is a computationally complex activity as an exponential amount of coalitions $(2^n - 1)$ have to be checked [3, 12]. In this paper we discuss an attempt to model the emotions of anger and gratitude so that they may be used as a basis to form coalitions.

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 81–87

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We propose that these relationships will reduce the computation costs placed on agents if coalition formation is highly complex, information about the environment is incomplete or uncertain, the robustness of other agent's proposed coalition is questionable, or there is a time-bound on coalition formation. We believe that modelling anger and gratitude to inform trust is applicable for use in such a task as trust is utilised when current conditions are highly complicated and uncertain [17] and it has also been proposed that the primary function of trust is to reduce complexity [8].

Therefore, we are not aiming to improve the *quality* of coalitions, instead we aim to reduce the associated computation cost by considering the roles that anger and gratitude can play in developing a mechanism of trust for use in coalition structure generation. In this paper we do *not* detail experimental results of testing this mechanism rather, we endeavour to: make the distinction between *emotional* and *rational* coalition formation processes clear, make the novel coalition formation process explicit and propose avenues for future work.

Section 2 of this paper introduces the background on coalition games, some initial discussion on the distinction between rationality and emotion and how emotions are modelled/used in artificial intelligence. Section 3 outlines the rational and emotional coalition formation methods used, how they are hybridised and a brief example illustrating how we propose the process works. Finally, section 4 concludes with an overview of the contributions made and some proposals for future work.

2 Background

Coalition formation takes place in an n-person cooperative game originally defined in [18] and is denoted: $\mathcal{G} = \langle N, v \rangle$ where N represents the set of agents ($N = \{1, ..., n\}$) and v is the *characteristic function* which assigns every coalition a real number that represents its payoff ($v(2^N) \rightarrow \mathbb{R}$). The *outcome* of an n-person cooperative game is a coalition structure $\{C^1, ..., C^k\}$ (where the individual coalitions are distinct and exhaustive) and a *payoff vector* that divides the gains of the coalition structure between all the agents. The payoff vector is fully denoted as: $x = (x_1, ...x_n) \in \mathbb{R}^n$ where x_i denotes the individual payoff for agent i and $x_i \geq 0$ for all $i \in N$ (see [3]). A payoff vector *satisfies* the agents of the system if it is said to be *stable* i.e. a payoff vector where no subset of agents in the system have an incentive to deviate from the current coalition structure.

Solutions of coalition games focus on what is mathematically optimal to do but as previously noted in section 1, the most mathematically rational solution is not the only consideration made in human societies. Such a solution may not be found especially if coalition formation is time constrained and/or attempted in an environment that is quite complex [16]. This paper seeks to outline a coalition formation method which simplifies the equilibrium computations required in such coalition games. As most scenarios where humans form coalitions e.g. the workplace, friendships, marriages etc. are not one-shot games (like the standard coalition game formalism) but iterative interactions, we will take inspiration from how emotions are formed and used in such iterative interactions so as to affect coalition formation. The coalition formation method that we are proposing here is an example of the "good enough, soon enough" design paradigm as we put no guarantees on if optimal solutions are found; this is left for future work.

Before continuing, we distinguish between the notions of *emotional* and *rational* coalition formation methods as understanding this difference is integral to understanding the approach outlined. As in [7], we use Axelrod's tournament [1] as a basis for the distinction. In Axelrod's tournament, some submitted strategies considered what action to perform in the

current round based upon an assessment of past/present/future payoffs. It is this payoff-based reasoning that we term as *rational* reasoning whereas *emotional* reasoning simply takes into consideration the *emotional disposition* of the agent towards another i.e. there is no explicit reasoning regarding past/present/future payoffs when identifying agents to form coalitions with. Emotional dispositions are concise histories that can take into account a multitude of interaction features but can be represented easily. These dispositions can be consulted quickly to not only determine whether or not to form a coalition with an agent but to also rank potential coalitions. Such an approach is inspired by the way emotions are used by Nawwab et. al to alter the preference ordering of actions when different emotions are activated [10]. We should make clear here that in no way at all do we intend for *emotion* to be interpreted as *irrational*, as is the usual dictum.

We model the emotions of anger and gratitude using the Ortony, Clore and Collins model (OCC) [11] as a basis. We take the view that anger and gratitude play a functional role, following [5] and use them to inform agents about whether to trust another. The exact implementation of this emotional approach is detailed in section 3.

3 The Coalition Formation Method

In this section we first outline the rational and emotional coalition formation processes in sections 3.1 and 3.2 before providing an outline of how we combine these two processes in section 3.3. Furthermore, in section 3.3 we present mathematical evidence asserting that such a hybrid technique is capable of buying advantages in computation time over the standard rational approach. Finally, we talk through a brief example illustrating how we propose the technique will work in section 3.4.

3.1 The Rational Coalition Formation Process

The rational coalition formation process finds an optimal coalition structure and a stable payoff vector for the system, i.e. it exclusively considers payoffs. The issue with this process is that it is computationally expensive: the asymptotically fastest algorithm to solve the coalition structure generation problem runs in $O(3^n)$ time [15]. For our research we will use the rational model detailed in [14], which is a distributed dialogue game that finds both an optimal coalition structure and a stable payoff vector. Theoretically however, any rational coalition structure generation model could be combined with the emotional approach outlined in sections 3.2 and 3.3.

In the model of [14], communication only occurs between agents when they offer a proposal to form a coalition. Proposals can be simply viewed as a three part tuple: $\langle i, C, x(C)\rangle$, where agent i proposes that coalition C forms with the coalition payoff vector denoted $x(C)$. If an agent in C cannot object to this coalition and payoff vector with a counter proposal then this coalition and payoff vector is said to be stable. Once a stable coalition for every agent in the system exists then a stable coalition structure also exists which entails the completion of the rational coalition formation process.

3.2 The Emotional Coalition Formation Process

The emotional coalition formation process enables an agent to restrict the state-space of coalitions it has to search based upon its emotional disposition towards others. This emotional disposition is informed by the performance of those agents in past coalitions. To

achieve this, each agent i is endowed with the following that are inspired by Reilly's model of emotion in [13]:

- Anger/gratitude variable: In accordance with the OCC we model opposite pairs of emotions so there is one variable that represents anger/gratitude. This variable is denoted by $AngGrt_j$ where $AngGrt$ is the current value of the variable and j indicates the agent that the variable applies to. An agent can have $AngGrt \cdot (N-1)$ variables where N is the total number of agents in the system ($N-1$ is used as anger/gratitude may not be felt towards the agent experiencing them). If the $AngGrt$ for i is negative towards another agent j then it can be inferred that i is angry with j and does not trust it. If the $AngGrt$ for i is positive towards another agent j then it can be inferred that i feels gratitude towards j and trusts it. If the $AngGrt$ of i towards another agent j is 0 then i is neither grateful or angry with j. Therefore, an agent may only be grateful, angry or neutral towards another, it may never activate any combination of these emotions in tandem with respect to the same agent.
- Anger/gratitude activation threshold: two constant values (Ang, Grt) that applies to an agent's $AngGrt$ variable. If i's $AngGrt_j \leq Ang$, the effect of anger is manifested in i towards j. If i's $AngGrt_j \geq Grt$, the effect of gratitude is manifested in i towards j. The values of these variables could be varied in order to make agents more/less trusting.
- Anger/gratitude effect: Prescribes what happens when an agent's $AngGrt$ value towards another agent is $\leq Ang$ or $\geq Grt$. Anger and gratitude have opposing effects: if i's $AngGrt_j \leq Ang$ then i will not include j in its coalition state-space search. Conversely, if $AngGrt_j \geq Grt$ then i will include j in its coalition state-space search. The behaviour of agents during the course of their interactions are not modified due to the activation of emotions, only the coalition state-space is affected.

For the purposes of this paper, the value of $AngGrt$ is altered according to whether or not the coalition succeeds or fails however, the inputs to this emotional disposition alteration are context-dependent and can vary. If an agent i joins a coalition C with another agent j then the following two situations may occur in context of this paper:

1. The coalition C succeeds – all agents $i \in C$ increase all $AngGrt_j$ variables (where $j \in C \backslash \{i\}$) by some amount.
2. The coalition C fails – all agents $i \in C$ decrease all $AngGrt_j$ variables (where $j \in C \backslash \{i\}$) by some amount.

The value of $AngGrt$ implies a notion of anger/gratitude intensity as it is possible that i's $AngGrt_j$ and $AngGrt_k$ variables may infer anger towards both agents but i is less angry with j than it is with k as $AngGrt_j$ may be equal to -30 whilst $AngGrt_k$ may equal -40. Variations in intensity create preference orderings as agent i will propose a coalition with the agent who has the highest $AngGrt$ value first. If the proposal is refused, the agent will then propose a coalition with the agent who has the next highest $AngGrt$ value and so on.

3.3 Hybridisation of Rational and Emotional Processes

Initially, all $AngGrt$ variables for each agent are set to 0 i.e. all agents are emotionally neutral towards all others. So to make a decision about who to form a coalition with, agents consult the rational coalition formation process detailed in section 3.1 *only*. However, in subsequent rounds, if any of agent i's $AngGrt$ values equal Ang or Grt then the emotion-based coalition formation approach is used in order to determine who i will work with, with

no input from the rational coalition formation approach unless all *AngGrt* variables for i's coalition choices are equal.

Agent i informs other agents of its anger/gratitude by communicating the tuple $\langle i, \{+, -, =\}, j\rangle$ where $+$ represents gratitude, $-$ represents anger and $=$ represents emotion neutrality. As stated, it is initially assumed that $\forall\, i, j \in N$ the tuple $\langle i, =, j\rangle$ holds.

Now for all agents $k \in N$, agent k will know that a proposal for a coalition has to include i and not j if $\langle i, -, j\rangle$ holds, which restricts the state space to search. Alternatively, both $\langle i, +, j\rangle$ and $\langle j, +, i\rangle$ have to hold to restrict the state space as both i and j have to want to be in a coalition together before it is fair to force them into one. If it is the case that i's *AngGrt* is negative/positive towards all agents then agent i will try to form a coalition with the agent(s) it is least angry with/most grateful to, respectively (explained in section 3.2).

The advantage of these restrictions are clear: consider a coalition search space of $2^n - 1$ coalitions (for n agents) and one anger constriction of $\langle i, -, j\rangle$ then there will be $(2^n - 1) - (2^{n-2})$ coalitions to check as 2^{n-2} is the amount of potential coalitions that any two agents share. Likewise, given the same coalition search space and one reciprocal gratitude relationship between agent i and j, then the amount of coalitions to search is reduced to $(2^n - 1) - (2^{n-1})$ as 2^{n-1} is the amount of potential coalitions for n agents that hold one of i and j but not the other. Table 1 shows a 4-person coalition game where there are $2^4 - 1$ possible coalitions that need to be checked. With the addition of one anger tuple $\langle 1, -, 2\rangle$ the underlined coalitions do not need to be searched. The underlined coalitions are equal to 2^{4-2} so the full amount of possible coalitions to search is: $(2^4 - 1) - (2^{4-2}) = 11$, which is a reduction from the full 15 used in the rational model. The benefit of this approach increases as more anger and gratitude relationships hold, especially if n becomes unmanageable for standard rational coalition structure generation techniques.

Table 1 State space reduction using emotional coalition formation process.

$\lvert C\rvert = \mathbf{1}$	$\lvert C\rvert = \mathbf{2}$	$\lvert C\rvert = \mathbf{3}$	$\lvert C\rvert = \mathbf{4}$
$\{1\}$	$\{3, 4\}$	$\{2, 3, 4\}$	$\underline{\{1, 2, 3, 4\}}$
$\{2\}$	$\{2, 4\}$	$\{1, 3, 4\}$	
$\{3\}$	$\{2, 3\}$	$\underline{\{1, 2, 4\}}$	
$\{4\}$	$\{1, 4\}$	$\underline{\{1, 2, 3\}}$	
	$\{1, 3\}$		
	$\underline{\{1, 2\}}$		

3.4 Example Process

When the *AngGrt* variables for multiple agents in a MAS begin to equal or surpass their *Ang*/*Grt* values emotions, a natural representation of these relationships, inspired by [2], is a directed graph with two arrow types:

- Pointed arrow heads from agent i to agent j for the tuple $\langle i, +, j\rangle$
- Flat arrow heads from agent i to agent j for the tuple $\langle i, -, j\rangle$

Figure 1 gives an example of such a directed graph for a MAS coalition game. The emotional dispositions asserted in this game are: $\langle 1, +, 2\rangle$, $\langle 2, +, 1\rangle$, $\langle 2, +, 5\rangle$, $\langle 3, +, 2\rangle$, $\langle 4, -, 2\rangle$, $\langle 5, -, 3\rangle$ and $\langle 5, +, 4\rangle$. If no arrow exists between two agents then these agents are emotionally indifferent to each other.

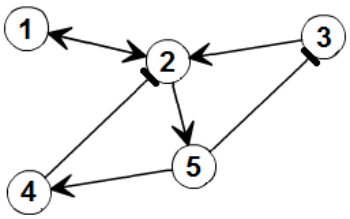


Figure 1 Directed graph denoting emotional dispositions in a 5 agent coalition game.

In this 5 agent game, the agents know that a 2-person coalition is at least needed to complete a task but a task gets easier with more agents. Let us assume that agent 1 makes the first proposal for a coalition (the mechanism which determines proposal orders is outside the scope of this paper). Agent 1 has to include agent 2 in its proposal as the agents have reciprocal positive emotions (so the current best coalition for agent 1 is $C_1 = \{1, 2\}$).

Agent 1 then has a choice as to the next agent to invite into the existing coalition and so considers both $C_2 = \{1, 2, 3\}$ and $C_3 = \{1, 2, 5\}$. Agent 4 is not considered as agent 2 is currently angry with agent 4 and therefore does not trust it, so agent 2 would reject the coalition of $\{1, 2, 4\}$. Agent 1 therefore has a choice between 2 agents: 3 and 5. As agent 1 is emotionally indifferent to both, the rational method is used to decide between the different coalitions. The best possible payoff for each coalition of agent 1's are: $x_1(C_3) > x_1(C_2) > x_1(C_3)$, so agent 1 deems C_3 to be the best coalition. Adding agent 5 to the proposal means that agent 3 should not be considered as agent 5 feels anger towards agent 3 and does not trust it. As no more agents can be added to the coalition, C_3 is then proposed, accepted and formed. Notice here that coalition C_3 was chosen by agent 1 out of a possible 2^{5-1} coalitions yet agent 1 only considered and compared 3 different possible coalition proposals (C_1, C_2 and C_3).

The acceptance of C_3 leaves the remaining two agents (3 and 4 who are indifferent to each other) to form a coalition if they want to complete a task. So, $C_4 = \{3, 4\}$ is also proposed, accepted and then formed, resulting in a coalition structure of: $\{\{C_3\}, \{C_4\}\}$.

4 Conclusion and Future Work

We have discussed the details of a hybrid coalition formation method that uses a previously established rational coalition formation process augmented with an emotionally-inspired coalition formation process. We have outlined our proposal for how this method will work in context of MAS and have outlined the benefits to computation costs that the method imparts. The modelling of anger and gratitude has also been made explicit and how these emotions are used to create a notion of trust between agents in MAS. Finally, we have outlined and discussed an example MAS in which this hybrid method facilitates a reduction in computational time with regards to forming coalitions.

A number of directions for future work have been outlined in the paper. The obvious continuation of this work would be to implement agents in a MAS who could use this hybrid coalition formation method. Ideally, this would be performed in context of a simulation test-bed so as to investigate whether the method outlined buys us reduced computation time as anticipated. Further questions may also be asked i.e. how scalable is the method and does this approach inadvertently produce coalitions of better quality?

Furthermore, it would be interesting to identify other variables that may affect the emotional disposition of an agent rather than just goal success/failure. Such variables could

include time taken to complete the goal specified, effort expended by other agents, consequences of the actions of another agent in context of goal achievement, shares of payoffs distributed etc. Such considerations could give rise to coalitions that are of better quality as more factors are taken into consideration.

Finally we may also consider the effects of varying anger/gratitude activation thresholds in order to create agents that are more/less trusting. This notion of *emotional characters* may be used to extend simulations so that we may identify those characters that are the most/least successful with respect to the quality of the coalitions formed.

References

1 Robert Axelrod. *The Evolution Of Cooperation.* Basic Books, Inc., 1984.

2 Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Coalitions of arguments: A tool for handling bipolar argumentation frameworks. *Intelligent Systems*, 25(1):83–109, 2010.

3 Georgios Chalkiadakis, Edith Elkind, and Michael Wooldridge. *Computational Aspects of Cooperative Game Theory.* Morgan & Claypool Publishers, 2011.

4 Jennifer R. Dunn and Maurice E. Schweitzer. Feeling and believing: The influence of emotion on trust. *Journal of Personality and Social Psychology*, 88:736–748, 2005.

5 Nico H. Frijda. *The Emotions.* Cambridge University Press, 1987.

6 David J. Lewis and Andrew Weigert. Trust as a social reality. *Social Forces*, 63:967–985, 1985.

7 Martyn Lloyd-Kelly, Katie Atkinson, and Trevor Bench-Capon. Emotion as an enabler of co-operation. In *4th International Conference on Agents and Artificial Intelligence, ICAART*, 2012.

8 Niklas Luhmann. *Trust and Power.* Chichester: Wiley, 1979.

9 Daniel J. McAllister. Affect and cognition-based trust as foundations for interpersonal cooperation in organizations. *The Academy of Management Journal*, 38:24–59, 1995.

10 Fahd Saud Nawwab, Trevor Bench-Capon, and Paul E. Dunne. Emotions in rational decision making. In *Lecture Notes in Computer Science*, volume 6057, pages 273–291. Springer, 2010.

11 Andrew Ortony, Gerald L. Clore, and Allan Collins. *The Cognitive Structure of Emotions.* Cambridge University Press, 1988.

12 Talal Rahwan. *Algorithms for Coalition Formation in Multi-Agent Systems.* PhD thesis, University of Southampton, 2007.

13 W. Scott Neal Reilly. *Believable Social and Emotional Agents.* PhD thesis, Carnegie Mellon University, 1996.

14 Luke Riley, Katie Atkinson, and Terry Payne. A dialogue game for coalition structure generation with self-interested agents. In *The Fourth International Conference on Computational Models of Argument, COMMA*, 2012.

15 Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111:209–238, 1999.

16 Leen-Kiat Soh and Costas Tsatsoulis. Satisficing coalition formation among agents. In *Proceedings of AAMAS*, 2002.

17 James D. Thompson. *Organizations in action: Social science bases in administrative theory.* New York: McGraw-Hill, 1967.

18 John von Neumann and Oskar Morgenstern. *The Theory of Games and Economic Behavior.* Princeton University Press, 1944.

19 Michael Wooldridge. *An Introduction to Multi-Agent Systems Second Edition.* John Wiley & Sons, 2009.

Bounded Model Checking for Linear Time Temporal-Epistemic Logic*

Artur Męski[1,2], Wojciech Penczek[1,3], and Maciej Szreter[1]

1 **Institute of Computer Science, Polish Academy of Sciences, Poland**
{meski,penczek,mszreter}@ipipan.waw.pl
2 **University of Łódź, Faculty of Mathematics and Computer Science, Poland**
3 **University of Natural Sciences and Humanities, Siedlce, Poland**

Abstract

We present a novel approach to the verification of multi-agent systems using bounded model checking for specifications in LTLK, a linear time temporal-epistemic logic. The method is based on binary decision diagrams rather than the standard conversion to Boolean satisfiability. We apply the approach to two classes of interpreted systems: the standard, synchronous semantics and the interleaved semantics. We provide a symbolic algorithm for the verification of LTLK over models of multi-agent systems and evaluate its implementation against MCK, a competing model checker for knowledge. Our evaluation indicates that the interleaved semantics can often be preferable in the verification of LTLK.

1998 ACM Subject Classification D.2.4 Software/Program Verification

Keywords and phrases model checking, multi-agent systems, temporal-epistemic logic, verification, interpreted systems

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.88

1 Introduction

It is often crucial to ensure that multi-agent systems (MAS) conform to their specifications and exhibit some desired behaviour. This can be checked in a fully automatic manner using model checking [4], which is one of the rapidly developing verification techniques. Model checking has been studied by various researchers in the context of MAS and different modal logics for specifying MAS properties [2, 6, 7, 10, 13, 14, 20, 21].

In the verification of multi-agent systems, the construction of the full, reachable state-space is often required. This exploration can lead to the state-space explosion, where the size of the model grows exponentially with the number of agents. Therefore, several approaches alleviating this problem have been proposed. One of them is *bounded model checking* (BMC) [1], in which only a portion of the original model, truncated up to some specific depth, is considered. This approach can be combined either with a translation of the verification problem to the propositional satisfiability problem (SAT) [10, 18] or with techniques based on *binary decision diagrams* (BDDs) [9].

In this paper we present a novel approach to verification of MAS by BDD-based bounded model checking for linear time temporal logic extended with the epistemic component (LTLK, also called CKL_n [7]). The systems are modelled by two variants of Interpreted Systems: standard (IS) [5] and interleaved ones (IIS) [12]. IIS restrict IS by enforcing asynchronous

* Partly supported by National Science Centre under the grant No. 2011/01/B/ST6/05317 and 2011/01/B/ST6/01477. A longer version of this paper appeared in the proceedings of LAM'12 [16].

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 88–94

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

semantics. This modifies the popular modelling approach for MAS by bringing the semantics known from verification of concurrent systems like networks of automata or variants of Petri nets. Our paper shows that the modelling approach has a very strong impact on the efficiency of verification. The experimental results exhibit that the IIS-based approach can greatly improve the practical applicability of the bounded model checking method for LTLK.

There has been already some intensive research on BMC for MAS, but mostly for the properties expressible in CTLK, based either on SAT [8, 18] or on BDDs [9]. A SAT-based verification method for the LTLK properties of MAS, modelled by IIS, was put forward in [19]. Our technical report [15] presents a BDD-based approach to verification of LTLK for IIS, while the SAT- and BDD-based approaches for IIS are compared in [17].

The rest of the paper is organised as follows. Sec. 2 provides the basic definitions and notations for LTLK and IS. Our method is described in Sec. 3. The last section contains the discussion of an experimental evaluation of the approach and the final remarks.

2 Preliminaries

2.1 Interpreted Systems

The semantics of *interpreted systems* [5] provides a setting to reason about MAS by means of specifications based on knowledge and linear or branching time. We begin by assuming a MAS to be composed of n agents[1] $\mathcal{A}$. We associate a set of *possible local states* L_i and *actions* Act_i to each agent $i \in \mathcal{A}$. We assume that the special action ϵ_i, called "null", or "silent" action of agent i belongs to Act_i; as it will be clear below the local state of agent i remains the same if the null action is performed. Also note we do not assume that the sets of actions of the agents are disjoint. We call $Act = \prod_{i \in \mathrm{A}} Act_i$ the set of all possible *joint actions*, i.e. tuples of local actions executed by agents. We consider a *local protocol* modelling the program the agent is executing. Formally, for any agent i, the actions of the agents are selected according to a *local protocol* $P_i : L_i \to 2^{Act_i}$. For each agent i, we define a relation $t_i \subseteq L_i \times Act \times L_i$, where $(l, (a_1, \ldots, a_n), l) \in t_i$ for each $l \in L_i$ if $a_i = \epsilon_i$. A *global state* $g = (g_1, \ldots, g_n)$ is a tuple of local states for all the agents corresponding to an instantaneous snapshot of the system at a given time. Given a global state $g = (g_1, \ldots, g_n)$ we denote by $l_i(g)$ the local component g_i of agent $i \in \mathcal{A}$ in g.

For each agent $i \in \mathcal{A}$, $\sim_i \subseteq G \times G$ is an *epistemic indistinguishability* relation over global states defined by $g \sim_i h$ if $l_i(g) = l_i(h)$. Further, let $\Gamma \subseteq \mathcal{A}$. The union of Γ's accessibility relations is defined as $\sim_\Gamma^E = \bigcup_{i \in \Gamma} \sim_i$. By $\sim_\Gamma^C$ we denote the transitive closure of $\sim_\Gamma^E$, whereas $\sim_\Gamma^D = \bigcap_{i \in \Gamma} \sim_i$.

A *global evolution* $\mathcal{T} \subseteq G \times Act \times G$ is defined as follows: $(g, a, h) \in \mathcal{T}$ iff there exists an action $a = (a_1, \ldots, a_n) \in Act$ such that for all $i \in \mathcal{A}$ we have $a_i \in P_i(l_i(g))$ and $(l_i(g), a, l_i(h)) \in t_i$. For $g, h \in G$ and $a \in Act$ s.t. $(g, a, h) \in \mathcal{T}$ we write $g \xrightarrow{a} h$. We assume that the global evolution relation $\mathcal{T}$ is total, i.e., for each $g \in G$ there exists $a \in Act$ and $h \in G$ such that $g \xrightarrow{a} h$.

An *infinite* sequence of global states and actions $\rho = g_0 a_0 g_1 a_1 g_2 \ldots$ is called a *path* originating from g_0 if there is a sequence of transitions from g_0 onwards, i.e., $g_i \xrightarrow{a_i} g_{i+1}$ for every $i \geq 0$. Any *finite* prefix of a path is called a *run*. By $length(\rho)$ we mean the number of the states of ρ if ρ is a run, and ω if ρ is a path. In order to limit the indices range of ρ which

[1] Note in the present study we do not consider the environment component. This may be added with no technical difficulty at the price of heavier notation.

can be a path or run, we define the relation $\trianglelefteq_\rho$. Let $\trianglelefteq_\rho \stackrel{def}{=} <$ if ρ is a path, and $\trianglelefteq_\rho \stackrel{def}{=} \leq$ if ρ is a run. A state g is said to be *reachable* from g_0 if there is a path or a run $\rho = g_0 a_0 g_1 a_1 g_2 \dots$ such that $g = g_i$ for some $i \geq 0$. The set of all the paths and runs originating from g is denoted by $\Pi(g)$. The set of all the paths originating from g is denoted by $\Pi^\omega(g)$.

▶ **Definition 1.** Given a set of propositions $\mathcal{PV}$ such that $\{true, false\} \subseteq \mathcal{PV}$, an *interpreted system* (IS), also called a *model*, is a tuple $M = (G, \iota, \mathcal{T}, \{\sim_i\}_{i\in\mathcal{A}}, \mathcal{V})$, where G is a set of global states, $\iota \in G$ is an initial (global) state such that each state in G is reachable from ι, $\mathcal{T}$ is the global evolution relation defined as above, and $V : G \to 2^{\mathcal{PV}}$ is a valuation function.

We define $\Pi = \bigcup_{g\in G} \Pi(g)$ to be the set of all the interleaved paths and runs originating from all states in G. By Π^ω we denote the set of all the paths of Π.

2.2 Interleaved Interpreted Systems

We define a restriction of interpreted systems, called *interleaved interpreted systems* in which global evolution function is restricted, so that every agent either executes a shared action or the null action. We assume that $\epsilon_i \in P_i(l)$, for any $l \in L_i$, i.e., we insist on the null action to be enabled at every local state. For each action $a \in \bigcup_{i\in\mathcal{A}} Act_i$ by $Agent(a) \subseteq \mathcal{A}$ we mean all the agents i such that $a \in Act_i$, i.e., the set of the agents potentially able to perform a. Then, the global evolution relation $\mathcal{T}$ is defined as before, but it is restricted by the following condition: if $(g, a, h) \in \mathcal{T}$ then there exists a joint action $a = (a_1, \dots, a_n) \in Act$, and an action $\alpha \in \bigcup_{i\in\mathcal{A}} Act_i \setminus \{\epsilon_1, \dots, \epsilon_n\}$ such that: $a_i = \alpha$ for all $i \in Agent(\alpha)$, and $a_i = \epsilon_i$ for all $i \in \mathcal{A} \setminus Agent(\alpha)$.

2.3 Syntax and Semantics of LTLK

▶ **Definition 2** (Syntax)**.** Let $\mathcal{PV}$ be a set of atomic propositions to be interpreted over the global states of a system, $p \in \mathcal{PV}$, $q \in \mathcal{A}$, and $\Gamma \subseteq \mathcal{A}$. Then, the syntax of LTLK is defined by the following BNF grammar: $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathrm{X}\varphi \mid \varphi \mathrm{U} \varphi \mid \varphi \mathrm{R} \varphi \mid \mathrm{K}_q\varphi \mid \overline{\mathrm{K}}_q\varphi \mid \mathrm{E}_\Gamma\varphi \mid \overline{\mathrm{E}}_\Gamma\varphi \mid \mathrm{D}_\Gamma\varphi \mid \overline{\mathrm{D}}_\Gamma\varphi \mid \mathrm{C}_\Gamma\varphi \mid \overline{\mathrm{C}}_\Gamma\varphi$.

The temporal operators U and R are named as usual, respectively, *until* and *release*; X is the next step operator. The epistemic operators K_q, $\mathrm{D}_\Gamma, \mathrm{E}_\Gamma$, and C_Γ [5] represent, respectively, knowledge of agent q, distributed knowledge in the group Γ, "everyone in Γ knows", and common knowledge among agents in Γ, whereas $\overline{\mathrm{K}}_q$, $\overline{\mathrm{D}}_\Gamma, \overline{\mathrm{E}}_\Gamma$, and $\overline{\mathrm{C}}_\Gamma$ are the corresponding dual.

The logic LTL is the sublogic of LTLK which consists only of the formulae built without the epistemic operators, whereas ELTL is a fragment of LTL where negation can be applied to propositions only. ELTLK is the existential fragment of LTLK, defined by the following grammar: $\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathrm{X}\varphi \mid \varphi \mathrm{U} \varphi \mid \varphi \mathrm{R} \varphi \mid \overline{\mathrm{K}}_q\varphi \mid \overline{\mathrm{E}}_\Gamma\varphi \mid \overline{\mathrm{D}}_\Gamma\varphi \mid \overline{\mathrm{C}}_\Gamma\varphi$.

▶ **Definition 3** (Semantics)**.** Given a model $M = (G, \iota, \mathcal{T}, \{\sim_q\}_{q\in\mathcal{A}}, \mathcal{V})$, where $\mathcal{V}(s)$ is the set of propositions that hold at s, let Π be a set of all the paths and runs of M, and $\rho(i)$ denote the i-th state of a path or run $\rho \in \Pi$, and $\rho[i]$ denote the path or run ρ with a designated formula evaluation position i, where $i \trianglelefteq_\rho length(\rho)$. Note that $\rho[0] = \rho$. The formal semantics of LTLK is defined recursively as follows:

$M, \rho[i] \models p$ iff $p \in \mathcal{V}(\rho(i))$,
$M, \rho[i] \models \neg\varphi$ iff $M, \rho[i] \not\models \varphi$,

$$
\begin{array}{lll}
M,\rho[i] \models \varphi_1 \wedge \varphi_2 & \text{iff} & M,\rho[i] \models \varphi_1 \text{ and } M,\rho[i] \models \varphi_2,\\
M,\rho[i] \models \varphi_1 \vee \varphi_2 & \text{iff} & M,\rho[i] \models \varphi_1 \text{ or } M,\rho[i] \models \varphi_2,\\
M,\rho[i] \models \mathrm{X}\varphi & \text{iff} & \mathit{length}(\rho) > i \text{ and } M,\rho[i+1] \models \varphi,\\
M,\rho[i] \models \varphi_1 \mathrm{U} \varphi_2 & \text{iff} & (\exists k \geq i)[M,\rho[k] \models \varphi_2 \text{ and } (\forall i \leq j < k)\ M,\rho[j] \models \varphi_1],\\
M,\rho[i] \models \varphi_1 \mathrm{R} \varphi_2 & \text{iff} & [\rho \in \Pi^{\omega}(\iota) \text{ and } (\forall k \geq i)\ M,\rho[k] \models \varphi_2]\\
 & & \text{or } (\exists k \geq i)[M,\rho[k] \models \varphi_1 \text{ and } (\forall i \leq j \leq k)\ M,\rho[j] \models \varphi_2],\\
M,\rho[i] \models \mathrm{K}_q\varphi & \text{iff} & (\forall \rho' \in \Pi^{\omega}(\iota))(\forall k \geq 0)[\rho'(k) \sim_q \rho(i) \text{ implies } M,\rho'[k] \models \varphi],\\
M,\rho[i] \models \overline{\mathrm{K}}_q\varphi & \text{iff} & (\exists \rho' \in \Pi(\iota))(\exists k \geq 0)[\rho'(k) \sim_q \rho(i) \text{ and } M,\rho'[k] \models \varphi],\\
M,\rho[i] \models \mathrm{Y}_\Gamma\varphi & \text{iff} & (\forall \rho' \in \Pi^{\omega}(\iota))(\forall k \geq 0)[\rho'(k) \sim_\Gamma^{\mathrm{Y}} \rho(i) \text{ implies } M,\rho'[k] \models \varphi],\\
M,\rho[i] \models \overline{\mathrm{Y}}_\Gamma\varphi & \text{iff} & (\exists \rho' \in \Pi(\iota))(\exists k \geq 0)[\rho'(k) \sim_\Gamma^{\mathrm{Y}} \rho(i) \text{ and } M,\rho'[k] \models \varphi],
\end{array}
$$

where $\mathrm{Y} \in \{\mathrm{D},\mathrm{E},\mathrm{C}\}$.

Moreover, an ELTLK formula φ holds in the model M, denoted $M \models_\exists \varphi$, iff $M,\rho \models \varphi$ for some path or run $\rho \in \Pi(\iota)$. The intuition behind this definition is that ELTLK is obtained by restricting the syntax of the epistemic operators while the temporal ones remain the same.

3 BDD-based Bounded Model Checking for ELTLK

To perform BMC of ELTLK using BDDs [4] we combine the standard approach for ELTL [3] with the method for the epistemic operators [20] in a similar manner to the solution for CTL* of [4] where the methods for CTL and LTL are combined into a method for CTL*.

Algorithm 1 Labelling algorithm

```
1:  M_c := M, φ_c := φ
2:  while γ(φ_c) ≠ 0 do
3:     pick ψ ∈ 𝒴(φ_c) such that γ(ψ) = 1
4:     for all g ∈ [[M_c, sub(ψ)]] do
5:        𝒱_{M_c}(g) := 𝒱_{M_c}(g) ∪ {p_{sub(ψ)}}
6:     end for
7:     ψ := ψ[sub(ψ) ← p_{sub(ψ)}]
8:     for all g ∈ [[M_c, ψ]] do
9:        𝒱_{M_c}(g) := 𝒱_{M_c}(g) ∪ {p_ψ}
10:    end for
11:    φ_c := φ_c[ψ ← p_ψ]
12: end while
13: return [[M_c, φ_c]]
```

Labelling algorithm. Given a model $M = (G,\iota,\mathcal{T},\{\sim_q\}_{q\in\mathcal{A}},\mathcal{V})$, a set $G_R \subseteq G$ of its reachable states, and an ELTLK formula φ, we compute the set $[\![M,\varphi]\!] = \{g \in G_R \mid M,g \models_\exists \varphi\}$ by reducing ELTLK to ELTL under the assumption that we have the algorithms for computing this set for each φ being an ELTL formula or in the form $\mathrm{Y}p$, where $p \in \mathcal{PV}$, and $\mathrm{Y} \in \{\overline{\mathrm{K}}_q, \overline{\mathrm{E}}_\Gamma, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{C}}_\Gamma\}$ (we use the algorithms from [3] and [20], respectively). In order to obtain this set, we construct a new model M_c together with an ELTL formula φ_c, and compute the set $[\![M_c,\varphi_c]\!]$, which is equal to $[\![M,\varphi]\!]$. Initially φ_c equals φ, which is an ELTLK formula, and we process the formula in stages to reduce it to an ELTL formula by replacing with atomic propositions all its subformulae containing epistemic operators. If $\varphi = \mathrm{Y}\psi$ is an ELTLK formula, by $\mathit{sub}(\varphi)$ we denote the formula ψ nested in the epistemic operator Y. We begin by choosing some epistemic subformula ψ of φ_c, which consists of exactly one epistemic operator (line 3), and process it in two stages. First, we modify the valuation function of

M_c (line 5) such that every state initialising some path or run along which $sub(\psi)$ holds is labelled with the new atomic proposition $p_{sub(\psi)}$, and we replace with the variable $p_{sub(\psi)}$ every occurrence of $sub(\psi)$ in ψ (line 7). In the second stage, we deal with the epistemic operators having in their scopes atomic propositions only. By modifying the valuation function of M_c (line 9) we label with a new variable p_ψ every state initialising some path or run along which the modified simple epistemic formula ψ holds. Similarly to the previous stage, we replace every occurrence of ψ in φ_c with p_ψ (line 11). In the subsequent iterations, we process every remaining epistemic subformulae of φ_c in the same way until there are no more nested epistemic operators in φ_c (line 2), i.e., we obtain an ELTL formula φ_c, and the model M_c with the appropriately modified valuation function. Finally, we compute the set of all reachable states of M_c that initialise at least one path or run along which φ_c holds (line 13).

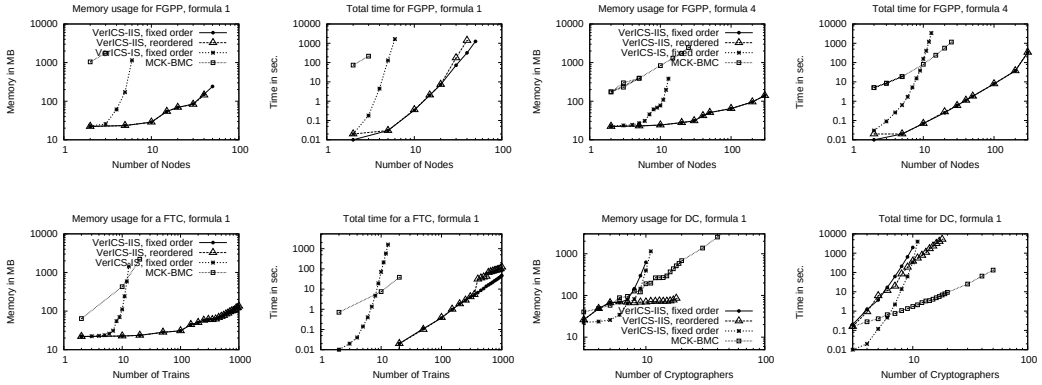
Algorithm 2 BMC algorithm

```
1:  Reach := {ι}, New := {ι}
2:  while New ≠ ∅ do
3:     Next := New_⇝
4:     if ι ∈ [[M|_Reach, φ]] then
5:        return true
6:     end if
7:     New := Next \ Reach
8:     Reach := Reach ∪ New
9:  end while
10: return false
```

BMC algorithm. Given a model M and an ELTLK formula φ, the algorithm checks if there exists a path or run initialised in the initial state ι along which φ holds. The algorithm starts with the set *Reach* of reachable states that initially contains only the state ι. With each iteration the verified formula is checked (line 4), and the set *Reach* is extended with new states reachable in one step from old states in *Reach* (line 8). The algorithm operates on submodels $M|_{Reach}$ generated by the set *Reach* (i.e., models restricted to contain only the states of *Reach*) to check if the initial state ι is in the set of states from which there is a path or run on which φ holds. The loop terminates if there is such a path or run in the obtained submodel, and the algorithm returns *true* (line 5). The search continues until no new states can be reached from the states in *Reach*. When we obtain the complete set of the reachable states, and a path or run from the initial state on which φ holds could not be found in any of the obtained submodels, the algorithm terminates returning *false*.

4 Experimental Evaluation

We have considered three scalable systems to evaluate the efficiency of our BDD-based BMC for LTLK: Faulty Generic Pipeline Paradigm (FGPP), Faulty Train Controller (FTC), and Dining Cryptographers (DC). The systems were modelled using two semantics, and the benchmarks were performed with several formulae. For the detailed descriptions of the benchmarks see [15]. Our method was implemented as two separate prototype modules of VerICS [11] for IS and IIS semantics (named VerICS-IS and VerICS-IIS, respectively). We have also compared our results with those obtained using MCK [6], another model checker for multi-agent systems, implementing standard IS semantics. Results for some of the performed benchmarks are included in the figures below.



Comparing algorithms for IS, in most cases MCK is better than VerICS-IS, but remains close when looking at the orders of magnitude. The reason for better performance of MCK may come from the fact that it is based on the translation to SAT, and SAT-based BMC does not need to store the whole examined part of the state space.

For most of the considered benchmarks the VerICS-IIS method is superior to the two IS approaches: MCK and VerICS-IS, sometimes even by several orders of magnitude. This can be observed especially in the case of FTC. However, in the case of FGPP and formula 3 with no epistemic modalities, MCK proved to be more efficient, but for the formula 4 containing the K operator, VerICS-IIS was superior. This can be justified by the fact that introducing epistemic modalities partitions the ELTL verification task into several smaller ones.

In the case of IIS, the reordering of the BDD variables does not cause any significant change of the performance in the case of FGPP and FTC, but for DC it reduces the memory consumption. Therefore, for IIS the fixed interleaving order we used can often be considered optimal. The penalty in the verification time to reorder the variables, in favour of reducing memory consumption, is also not significant and can be worth the tradeoff. However, in the case of IS the performance did not change, thus we include only the results for the fixed order of the variables for VerICS-IS.

It is important to note that from our comparison of [17] it follows that in the case of IIS, the general performance of BDD-based approach is superior to the SAT-based one. Therefore, we can conclude now that BMC for LTLK is less efficient for IS when comparing with IIS. This could be explained by the different structure of the state space, which for IS is more dense, i.e., more states are explored at every iteration of the BMC algorithm. The case of DC shows that this factor can be more important than the lengths of the counterexamples, which can be shorter for IS, or may even be of constant length when scaling the system.

The experimental results show that the approach based on the interleaved interpreted systems can greatly improve the practical applicability of the bounded model checking method. Although, we have tested only properties of LTLK, we suspect this to also be true for similar specification formalisms, e.g., CTLK.

5 Final Remarks

In this paper, we have presented a BDD-based method for bounded model checking of LTLK over models of multi-agent systems. We evaluated the methodology in two different settings: interleaved interpreted systems and synchronous interpreted systems. The results are preliminary and the comparison is by no means complete. It ignores the fact that for some formulae the choice of the semantics influences the existence of a witness in the model.

References

1 A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Proc. of the 5th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'99)*, volume 1579 of *LNCS*, pages 193–207. Springer-Verlag, 1999.

2 R. Bordini, M. Fisher, C. Pardavila, W. Visser, and M. Wooldridge. Model checking multi-agent programs with CASP. In *Proc. of CAV'03*, volume 2725 of *LNCS*, pages 110–113. Springer-Verlag, 2003.

3 E. Clarke, O. Grumberg, and K. Hamaguchi. Another look at LTL model checking. In *Proc. of CAV'94*, volume 818 of *LNCS*, pages 415–427. Springer-Verlag, 1994.

4 E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

5 R. Fagin, J. Y. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, 1995.

6 P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In *Proc. of the 16th Int. Conf. on Computer Aided Verification (CAV'04)*, volume 3114 of *LNCS*, pages 479–483. Springer-Verlag, 2004.

7 W. van der Hoek and M. Wooldridge. Model checking knowledge and time. In *Proc. of SPIN'02*, volume 2318 of *LNCS*, pages 95–111. Springer-Verlag, 2002.

8 X. Huang, C. Luo, and R. van der Meyden. Improved bounded model checking for a fair branching-time temporal epistemic logic. In *Proc. of 6th Int. Workshop on Model Checking and Artificial Intelligence 2010*, LNAI. Springer, 2011.

9 A. V. Jones and A. Lomuscio. Distributed bdd-based bmc for the verification of multi-agent systems. In *Proc. of AAMAS'10*, pages 675–682, 2010.

10 M. Kacprzak, A. Lomuscio, and W. Penczek. From bounded to unbounded model checking for temporal epistemic logic. *Fundam. Inform.*, 63(2-3):221–240, 2004.

11 M. Kacprzak, Wojciech Nabiałek, A. Niewiadomski, W. Penczek, A. Półrola, M. Szreter, B. Woźna, and A. Zbrzezny. Verics 2006 - a model checker for real-time and multi-agent systems. In *Proc. of the Int. Workshop on Concurrency, Specification and Programming (CS&P'07)*, pages 345–356. Warsaw University, 2007.

12 A. Lomuscio, W. Penczek, and H. Qu. Partial order reduction for model checking interleaved multi-agent systems. In *Proc. of AAMAS'10*, pages 659–666, 2010.

13 R. van der Mayden and K. Su. Symbolic model checking the knowledge of the dining cryptographers. In *Proc. of CSFW-17*, pages 280–291. IEEE Computer Society, June 2004.

14 R. van der Meyden and N. V. Shilov. Model checking knowledge and time in systems with perfect recall. In *Proc. of FSTTCS'99*, volume 1738 of *LNCS*, pages 432–445. Springer-Verlag, 1999.

15 A. Męski, W. Penczek, and M. Szreter. Bounded model checking linear time and knowledge using decision diagrams. In *Proc. of CS&P'11*, pages 363–375, 2011.

16 A. Męski, W. Penczek, and M. Szreter. BDD-based bounded model checking for LTLK over two variants of interpreted systems. In *Proc. of LAM'12*, pages 35–49, 2012.

17 A. Męski, W. Penczek, M. Szreter, B. Woźna-Szcześniak, and A. Zbrzezny. Bounded model checking for knowledge and linear time. In *Proc. of AAMAS'12*. IFAAMAS Press, 2012.

18 W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundam. Inform.*, 55(2):167–185, 2003.

19 W. Penczek, B. Woźna-Szcześniak, and A. Zbrzezny. Towards SAT-based BMC for LTLK over interleaved interpreted systems. In *Proc. of CS&P'11*, pages 565–576, 2011.

20 F. Raimondi and A. Lomuscio. Automatic verification of multi-agent systems by model checking via OBDDs. *Journal of Applied Logic*, 5(2):235–251, 2007.

21 K. Su, A. Sattar, and X. Luo. Model checking temporal logics of knowledge via OBDDs. *The Computer Journal*, 50(4):403–420, 2007.

A compositional model to characterize software and hardware from their resource usage

Davide Morelli and Antonio Cisternino

Computer Science Department, University of Pisa
Largo B. Pontecorvo 3, Italy
(morelli|cisterni)@di.unipi.it

Abstract

Since the introduction of laptops and mobile devices, there has been a strong research focus towards the energy efficiency of hardware. Many papers, both from academia and industrial research labs, focus on methods and ideas to lower power consumption in order to lengthen the battery life of portable device components. Much less effort has been spent on defining the responsibility of software in the overall computational system's energy consumption.

Some attempts have been made to describe the energy behaviour of software, but none of them abstract from the physical machine where the measurements were taken. In our opinion this is a strong drawback because results can not be generalized. We propose a measuring method and a set of algebraic tools that can be applied to resource usage measurements. These tools are expressive and show insights on how the hardware consumes energy (or other resources), but are equally able to describe how efficiently the software exploits hardware characteristics. The method is based on the idea of decomposing arbitrary programs into linear combinations of benchmarks of a test-bed without the need to analyse a program's source code by employing a black box approach, measuring only its resource usage.

1998 ACM Subject Classification D.2.8 Metrics, D.4.8 Performance, B.8.2 Performance Analysis and Design Aids

Keywords and phrases Performance, Metrics, Energy consumption

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.95

1 Introduction

Energy consumption is a global concern; as the Environmental Protection Agency of the U.S. stated in a report [8] dated 2007, the energy consumed by data centres and servers alone can account for 1.5% of the global energy use, and is doubling every five years.

The adoption rate of portable devices raised the attention towards energy efficiency of hardware components (in order to lengthen battery life) and network protocols. Much less effort has been spent on defining the responsibility of software in the overall computational system's energy consumption, none of them abstract from the physical machine where the measures are taken. In our opinion this is a strong drawback because results can not be generalized.

We propose a measuring method and a set of algebraic tools that can be applied to resource usage measurements (energy consumption and completion time being just two instances of resource usage). These tools are expressive and show insights on how the hardware consumes energy (or other resources). They are also able to describe how efficiently the software exploits the hardware characteristics.

Typically, measurement techniques proposed in the literature aim to break down the energy consumption to atomic components, associating an average cost to every instruction [16, 15,

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 95–101

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

9, 2, 17]. We believe that a different approach should be used, as modern computational systems tend towards complex systems. Hardware started developing parallelism when the CPU frequency reached its cap, and software systems are becoming increasingly complex. The result of this process is that the resource usage of a single instruction is hardly predictable and depends on the execution context. For example, the time required to load a location depends on both the program's memory access pattern, the hardware characteristics, and the memory used by the other processes running on the same computational system, leading to almost *non deterministic* results. For this reasons we have chosen to follow a black box approach, measuring the resource usage of the software running on a computational system as a whole, following the approach proposed in [11], instead of trying to profile the software [9], simulating the execution of algorithms on modified virtual machines or power level performance simulators [2, 17]. This is because these approaches either require source code access or are only feasible for simple architectures, where a cycle accurate simulator is possible and for relatively small programs. A black box approach measures the software as a whole without trying to break down the energy consumption of single instructions. This approach is the most simple to implement, does not need modifications to the operative system and works with a simple ammeter with the simplest possible approach: the current is measured in AC from the wall outlet [14, 17, 4].

Metrics for software similarity are very interesting because they allow us to predict the behaviour of programs using measurements of similar programs, and allow their characterization. Yamamoto proposes a metric of similarity based on source code analysis in [18]. For the scope of our research, we are interested in methods that do not require access to the source code because we want to be able to characterize software as a black box.

In a 1992 paper [12] we found an approach that was particularly inspiring for our work: a model was proposed to characterize both hardware and software. The overall completion time of a program was modelled as a linear decomposition of abstract operations. Our model is very similar to their with respect to the linear composition model. Nevertheless, there are many important differences:

- they use abstract operations as computational patterns (e.g. *add*, *store*, *divide*, etc.); we allow more articulate entities, long combinations of instructions
- the program analysis is performed by means of static analysis and instrumentation of the source code; we don't require the source code to analyse the program
- their model focuses on completion time solely; our model is capable of describing the usage of every measurable resource, we are most interested in completion time and energy consumption, but could be applied to any other metric (i.e. memory usage, CPU time, etc.)

[17] also proposed an energy characterization model for both hardware and software.

In [1], the use of performance counters leverages the characterization of software, a technique that is becoming a de facto standard [5, 10, 7, 6]. Benchmarks are analyzed, PCA is used to reduce the solution space and clustering techniques are used to identify families of programs and to find a representative workload for a certain task.

Two other key concepts are the idea that the environment where the program is run must be taken into account [3] and the need to find a model capable of offering results resilient with change of hardware: Sherwood [13] characterized software with a model consistent with the change of architecture; he proposed a high level approach (not at instruction level), but he did not focus on energy consumption.

2 A compositional model

Programs are composed of instructions that once executed affect the resource consumption of the system. There are many different kinds of computational resources a program can consume (CPU time, memory, network, etc.). We define *computational pattern* a sequence of instructions that expose a peculiar resource usage, that is subject to change as we change the computational system where the pattern is executed on, e.g. on a processor family FPU operation may consume more energy to complete with respect to a different class of processors.

In our model we assume that actual programs can be seen as composed of computational patterns, i.e. a matrix multiplication algorithm will read data from memory (showing a peculiar memory read pattern with cache hit and miss), perform FPU then write the result back to memory.

A computational pattern will have a different resource usage on each computational system, e.g. the same memory pattern of data read could rise to a much lower number of cache miss if a processor is capable of predicting the pattern and prefetching data. The composition of a program from the point of view of the computational patterns does not change when the program is run on a different computational system, but the resource usage behaviour will change because the computational pattern the program is composed of will have a different resource usage profile.

Therefore, we chose a set of synthetic benchmarks as our *test-bed*, where every benchmark is intended to capture a particular computational pattern that we expect to find in different quantities in every program we intend to analyse, with zero being a legitimate quantity.

3 Linear algebra model

We define the *measurement matrix* a computational system S as a $\mathbb{R}^{m\times n}$ matrix where n is the number of benchmarks in our test-bed and m is the number of attributes (resource usage) we measure for each program. Each one of these matrices holds the knowledge we have about a particular computational system. The i^{th} column of $\mathbf{M}$ shows the resource usage of the i^{th} benchmark of our test-bed. The j^{th} cell of that column holds the measurement of the resource usage of the j^{th} attribute (e.g. CPU time, cache hit, etc.).

When we want to decompose a program p using the benchmarks of our test-bed as the building block we measure the resources usage of p running on S and we build a vector μ_p with those measures. The j^{th} element of μ_p holds the measurement of the resource usage of the j^{th} attribute; μ_p is like a column of $\mathbf{M}$ but is composed of measurements of a program that is not part of the test-bed. Now consider the following linear system:

$$\mathbf{M}\cdot\mathbf{s}_p = \mu_p \tag{1}$$

We call $\mathbf{s}_p$ the *split-up* of p, it holds a decomposition of p using the benchmarks of our test-bed as building blocks.

Standard vector algebra can be used to analyse and interpret measures, splitups and programs. We can analyse vectors using vector norm:

$$\|v\| = \sqrt{\sum_{i=1}^{n}(v_i)^2} \tag{2}$$

and vector similarity:

$$cos(\theta) = \frac{v_1 \cdot v_2}{\|v_1\|\|v_2\|} \tag{3}$$

3.1 Measurements space

The columns of M can be seen as vectors in an m dimensional vector space that we call the *measurements space*. The position of the i^{th} vector shows the resource usage of the i^{th} benchmark of the test-bed. More generally speaking, the μ_p vector shows the resource usage of the program p in a particular system S.

The norm can be used to get an insight on the overall resource usage of a benchmark, i.e. more resource demanding benchmarks will have a higher norm than less resource demanding ones.

Vector similarity will tell us how similar two programs are: if the angle between the vectors is small it means that they may use more or less resources in absolute terms (have different norms), but their resource usage behaviour is similar.

3.2 Splitup space

Splitups can be seen as vectors in a n dimensional vector space that we call the *splitup space*. The position of a vector in this space shows the composition of the program using the benchmarks as building blocks.

When comparing the splitup vectors of two programs we can say that the one with a higher norm is the more resources demanding. If the vector similarity is very close to 1 but the norms are different we are probably looking at the same program running with different input sizes, e.g. if p_1 and p_2 are the same sorting algorithm with p_1 running on half the array size of p_2 we'll probably see $\|\mathbf{s}_{p_2}\| = 2\|\mathbf{s}_{p_1}\|$ and $\frac{\mathbf{s}_{p_1} \cdot \mathbf{s}_{p_2}}{\|\mathbf{s}_{p_1}\|\|\mathbf{s}_{p_2}\|} = 1$

When a program is run on different input sizes the balance of the computational patterns used may vary, e.g. the cache hit ratio could grow logarithmically while the FPU usage may grow linearly. Analysing how the splitup of a program changes with the input size is highly informative of the program structure. When the splitup does not change with the input size we call the program *uniform*; and if it changes, we call it *non uniform*.

If the test-bed is well formed the splitup of a program has to be the same when we run it on different computational systems. If it is different it means that this program is capturing a resource usage behaviour not captured by any benchmark in the test-bed, in other words this program contains an unknown computational pattern, therefore it should be added to our test-bed.

3.3 Benckmark space

$\mathbf{M}_S$ is the $\mathbf{M}$ of a system S. Its rows can be seen as vectors in a n^{th} dimensional space that we call the *benchmark space*. We can see how resource usage changes when we change computational system from S_a to S_b analysing the position of the i^{th} row of $\mathbf{M}_{S_a}$ (where i is the index of the resource we are interested in) against the position of the i^{th} row of $\mathbf{M}_{S_b}$ in the benchmark space.

E.g. when the norm of the energy consumption vector of S_1 is higher than the norm of the energy consumption vector of S_2 it means that S_1 is (generally speaking) less energy efficient than S_2. If the angle between their completion time vectors is small it means that S_1 and S_2 have a similar architecture and probably one is just more efficient than the other (i.e.

a newer machine). If the angle is large it means that the systems have a different architecture that makes some of the benchmarks in the test-bed more efficient than others; the direction of the difference between the vectors tells us how S_1 is different from S_2.

4 Real data

Measuring resource attribute will usually involve error, i.e. the accuracy of the measuring tool, sampling frequency, etc. Equation 1 could be not solvable and has to be rewritten in order to minimise a norm of the error vector:

$$\epsilon = |\mathbf{M} \cdot \mathbf{s}_p - \mu_p| \tag{4}$$

If we use a Manhattan norm instead of an Euclidean norm, this is a linear programming problem that can be solved using the simplex algorithm.

We want all the elements of $\mathbf{s}_p$ to be non negative numbers, because each of them expresses an estimation of the number of iterations of the respective benchmark, as present in p. The benchmark space is therefore not a vector space but a convex cone. This limitation does not change the approach needed to find the splitup, since we just need to add a few conditions to the simplex.

Being the benchmark space a convex cone, the number of vectors (the benchmark in the test-bed) that form a basis is not generally known, but the process of selection of the benchmarks in the test-bed can be incremental and automatic: if the splitup of a program falls within the convex cone it means that it can be expressed as linear decomposition of known computational patterns, if it falls outside the cone it means that it should be added to the test-bed, widening the range of programs that can be expressed algebraically.

We can choose the level of detail we want to get with the decomposition of programs. I.e., we might want to have a computational pattern for every major memory read pattern or just a general one. In the former case we would be able to discriminate how the program uses memory, but we would need a lot of experimental data to solve the system. In the latter, we would need few experimental data but might only see a raw estimate of the program's behaviour. The number of rows of the measurement matrix needs to be larger than the number of columns, which means that we need to measure at least as many resources as the number of the benchmarks in the test-bed. This could be difficult if we want to have a large test-bed, in which case we could create a new measurement matrix with more rows just merging measurement matrices of multiple systems.

5 Experimental data

As an example we present data of a preliminary test: we measured the completion time and the energy consumption of a small set of programs running on a desktop computer equipped with a CoreDuo processor with 2MB L2 cache and 1 GB RAM (from now on referred to as S). We prepared two synthetic benchmarks: *cpu* is a simple *add* assembler instruction executed 10^6 times; *mem* is a program that sums a fixed number of random locations from a large array. We used *cpu* and *mem* as our test-bed and measured mergesort (from now on referred to as p) sorting arrays of different sizes (1M, 2M, 4M, 8M, 16M, 32M).

	cpu	mem	p(1M)	p(2M)	p(4M)	p(8M)	p(16M)	p(32M)
time	2.14 s	7.26 s	0.22 s	0.33 s	0.67 s	1.39 s	2.85 s	5.79 s
energy	81.46 J	304.00 J	8.60 J	13.20 J	27.49	58.29 J	121.79 J	254.82 J

$\mathbf{M}_S$ is composed of the first two columns of the above table and the measurement vectors for mergesort at various input sizes are:

$$\mu_{p(1M)} = \begin{pmatrix} 0.22\,\mathrm{s} \\ 8.60\,\mathrm{J} \end{pmatrix} \quad \mu_{p(2M)} = \begin{pmatrix} 0.33\,\mathrm{s} \\ 13.20\,\mathrm{J} \end{pmatrix} \quad \mu_{p(4M)} = \begin{pmatrix} 0.67\,\mathrm{s} \\ 27.49\,\mathrm{J} \end{pmatrix}$$
$$\mu_{p(8M)} = \begin{pmatrix} 1.39\,\mathrm{s} \\ 58.29\,\mathrm{J} \end{pmatrix} \quad \mu_{p(16M)} = \begin{pmatrix} 2.85\,\mathrm{s} \\ 121.79\,\mathrm{J} \end{pmatrix} \quad \mu_{p(32M)} = \begin{pmatrix} 5.79\,\mathrm{s} \\ 254.82\,\mathrm{J} \end{pmatrix}$$

The resulting splitup vectors (calculated minimizing formula 4 using the simplex algorithm) are:

$$\mathbf{s}_{p(1M)} = \begin{pmatrix} 0.075118 \\ 0.008161 \end{pmatrix} \quad \mathbf{s}_{p(2M)} = \begin{pmatrix} 0.075862 \\ 0.023093 \end{pmatrix} \quad \mathbf{s}_{p(4M)} = \begin{pmatrix} 0.069347 \\ 0.071845 \end{pmatrix}$$
$$\mathbf{s}_{p(10M)} = \begin{pmatrix} 0 \\ 0.248125 \end{pmatrix} \quad \mathbf{s}_{p(20M)} = \begin{pmatrix} 0 \\ 0.528191 \end{pmatrix} \quad \mathbf{s}_{p(30M)} = \begin{pmatrix} 0 \\ 0.780954 \end{pmatrix}$$

The splitup vectors show how quickly mergesort gets dominated by memory usage as the input size grows. This is expected because as the array grows it will not fit into cache and a lot of cache miss will occur, therefore most of the time and energy will be spent accessing memory.

6 Conclusion

We have presented a method to decompose arbitrary programs into linear combinations of benchmarks of a test-bed by employing a black box approach, measuring only its resource usage, without the need to analyse a program's source code. Valid metrics of resource usage are both performance counters and energy consumption (or completion time). Performance counters can therefore be used to build a model capable of predicting the energy consumption (or completion time) of the same program on a different computational system. The same method also gives us useful information about what the differences between computational systems are, thereby showing which computational patterns consume more resources. We intend to apply this method to heterogeneous computing (CPU/GPU), virtual machines and cloud systems to provide realtime analysis and forecasting of energy consumption (as well as completion time) of software, without prior knowledge of its source code.

References

1 Jan Lodewijk Bonebakker. Finding representative workloads for computer system design. Technical report, Mountain View, CA, USA, 2007.

2 David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th annual international symposium on Computer architecture*, ISCA '00, pages 83–94, New York, NY, USA, 2000. ACM.

3 Fay Chang, Keith Farkas, and Parthasarathy Ranganathan. Energy-driven statistical sampling: Detecting software hotspots. In Babak Falsafi and T. Vijaykumar, editors, *Power-Aware Computer Systems*, volume 2325 of *Lecture Notes in Computer Science*, pages 105–108. Springer Berlin / Heidelberg, 2003.

4 A. Cisternino, P. Ferragina, D. Morelli, and M. Coppola. Information processing at work: On energy-aware algorithm design. In *Green Computing Conference, 2010 International*, pages 407 –415, aug. 2010.

5 Matthew Curtis-Maury, James Dzierwa, Christos D. Antonopoulos, and Dimitrios S. Nikolopoulos. Online power-performance adaptation of multithreaded programs using hardware event-based prediction. In *Proceedings of the 20th annual international conference on Supercomputing*, ICS '06, pages 157–166, New York, NY, USA, 2006. ACM.

6 Evelyn Duesterwald, Calin Cascaval, and Sandhya Dwarkadas. Characterizing and predicting program behavior and its variability. In *Proceedings of the 12th International Conference on Parallel Architectures and Compilation Techniques*, PACT '03, pages 220–, Washington, DC, USA, 2003. IEEE Computer Society.

7 Lieven Eeckhout, Hans Vandierendonck, and Koen De Bosschere. Workload design: Selecting representative program-input pairs. *Parallel Architectures and Compilation Techniques, International Conference on*, 0:83, 2002.

8 R. Brown et al. Report to congress on server and data center energy efficiency: Public law 109-431, 2008.

9 Jason Flinn and M. Satyanarayanan. Powerscope: A tool for profiling the energy usage of mobile applications. In *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, WMCSA '99, pages 2–, Washington, DC, USA, 1999. IEEE Computer Society.

10 Aashish Shreedhar Phansalkar. *Measuring program similarity for efficient benchmarking and performance analysis of computer systems*. PhD thesis, Austin, TX, USA, 2007. AAI3285977.

11 Suzanne Marion Rivoire. *Models and metrics for energy-efficient computer systems*. PhD thesis, Stanford, CA, USA, 2008. AAI3313649.

12 Rafael H. Saavedra and Alan J. Smith. Analysis of benchmark characteristics and benchmark performance prediction. *ACM Trans. Comput. Syst.*, 14:344–384, November 1996.

13 Tomothy Sherwood, Erez Perelman, Greg Hamerly, and Brad Calder. Automatically characterizing large scale program behavior. *SIGOPS Oper. Syst. Rev.*, 36:45–57, October 2002.

14 Amit Sinha and Anantha P. Chandrakasan. Jouletrack - a web based tool for software energy profiling. In *In Design Automation Conference*, pages 220–225, 2001.

15 Stefan Steinke, Markus Knauer, Lars Wehmeyer, and Peter Marwedel. An accurate and fine grain instruction-level energy model supporting software optimizations. In *in Proc. Int. Wkshp Power and Timing Modeling, Optimization and Simulation (PATMOS*, 2001.

16 Vivek Tiwari, Sharad Malik, and Andrew Wolfe. Power analysis of embedded software: a first step towards software power minimization. In *Proceedings of the 1994 IEEE/ACM international conference on Computer-aided design*, ICCAD '94, pages 384–390, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

17 N. Vijaykrishnan, M. J. Irwin, H. S. Kim, and W. Ye. Energy-driven integrated hardware-software optimizations using simplepower. pages 95–106, 2000.

18 Tetsuo Yamamoto, Makoto Matsushita, Toshihiro Kamiya, and Katsuro Inoue. Measuring similarity of large software systems based on source code correspondence. In Frank Bomarius and Seija Komi-Sirviö, editors, *Product Focused Software Process Improvement*, volume 3547 of *Lecture Notes in Computer Science*, pages 179–208. Springer Berlin / Heidelberg, 2005.

Integration of Temporal Abstraction and Dynamic Bayesian Networks in Clinical Systems. A preliminary approach

Kalia Orphanou[1], Elpida Keravnou[2], and Joseph Moutiris[3]

1 **Department of Computer Science, University of Cyprus, Nicosia, Cyprus korfan01@cs.ucy.ac.cy**
2 **Cyprus University of Technology, Limassol, Cyprus rector@cut.ac.cy**
3 **Department of Cardiology, Nicosia, General Hospital, Cyprus moutiris@ucy.ac.cy**

Abstract

Abstraction of temporal data (TA) aims to abstract time-points into higher-level interval concepts and to detect significant trends in both low-level data and abstract concepts. TA methods are used for summarizing and interpreting clinical data. Dynamic Bayesian Networks (DBNs) are temporal probabilistic graphical models which can be used to represent knowledge about uncertain temporal relationships between events and state changes during time. In clinical systems, they were introduced to encode and use the domain knowledge acquired from human experts to perform decision support.

A hypothesis that this study plans to investigate is whether temporal abstraction methods can be effectively integrated with DBNs in the context of medical decision-support systems. A preliminary approach is presented where a DBN model is constructed for prognosis of the risk for coronary artery disease (CAD) based on its risk factors and using as test bed a dataset that was collected after monitoring patients who had positive history of cardiovascular disease. The technical objectives of this study are to examine how DBNs will represent the abstracted data in order to construct the prognostic model and whether the retrieved rules from the model can be used for generating more complex abstractions.

1998 ACM Subject Classification I.2.1 Applications and Expert Systems

Keywords and phrases temporal abstraction, medical prognostic models, dynamic Bayesian network, coronary artery disease

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.102

1 Introduction

Advances in the field of Artificial Intelligence led to the development of intelligent clinical data analysis systems, that are designed to provide computer-based support in medical tasks by automating, for example, diagnostic reasoning. In general, the purpose of medical data analysis systems is to aid care providers reach the best possible decisions for any patient, to help them understand what the possible consequences of their decisions/actions are and if necessary to take corrective actions in a short time interval.

Temporal abstraction (TA) [16] abstracts time-point based data into higher-level, interval based concepts under a given context. Abstraction of time-oriented clinical data aims to close the gap between general medical knowledge and specific patient data. Medical knowledge is expressed in a general form (association rules, patient management protocols) while patient data are specific (history of patients, results of laboratory and physical examinations). The

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 102–108

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

derived abstracted concepts are useful for different tasks such as decision making, therapy planning and summarization of a patient's record.

According to Bellazzi [1], time-stamped entities are called events and their abstract representation, given by TAs as sequences of intervals, are called episodes. The TA task can be divided into two subtasks, basic and complex abstractions. Basic temporal abstractions are state and trend abstractions which abstract events (time-stamped data) within episodes. Complex temporal abstractions abstract episodes (intervals) into other episodes. The aim of state abstraction is to derive maximal intervals over which there is no change in the state of some parameter. The aim of trend abstraction is to derive the significant changes and the rates of change in the progression of some parameters e.g stable, increasing, decreasing.

A Bayesian Network (BN) is an acyclic graph that represents a joint probability distribution over a set of random variables. It consists of two components, a directed acyclic graph and a probability distribution. Nodes on the graph represent the variables and edges represent the direct dependencies between the variables. BNs can make predictions or give explanations by computing the conditional probability table of each variable. BNs have been introduced as a knowledge representation method to encode and use the domain knowledge acquired from human experts in automated reasoning systems to perform diagnostic, predictive and explanatory tasks. They can represent knowledge even in cases of missing data or uncertain information.

Dynamic Bayesian Networks (DBNs) [3] are a temporal extension of standard BNs that are able to model stochastic processes. They utilize a representation of a dynamic process via a set of stochastic variables in a sequence of time-slices. More precisely, a DBN is a network with the repeated structure of a BN for each time slice over a certain interval. Consequently, a DBN is a tuple (B1, B2), where B1 is a Bayesian Network that represents the prior distribution for the variables in the first time slice and B2 represents the transition model for the variables in two consecutive time slices. Relations between variables are divided into two types: transitional relations represent dependencies between variables between different time slices and local relations represent dependencies among variables in the same time slice. DBNs are usually assumed that they use the Markovian property: conditional probability distribution of each variable at time t, for all t >1, depends only on the parents from the same time slice or from the previous time slice but not from earlier time slices.

This paper is organized as follows. Section 2 describes the preliminary approach of integrating these two areas under the domain of CAD. Section 3 describes related work. Section 4 discusses the potential advantages of this integration and future improvements of this approach and and the paper ends with closing remarks.

2 Preliminary Approach

2.1 Data Description

Coronary artery disease (CAD) is one of the major causes of disability in adults as well as one of the main causes of death in the developed countries. The dataset was collected after 4 years of monitoring patients (2009–2012) under the supervision of the participating cardiologist (Dr. J. Moutiris) at the Cardiology Clinics of Nicosia General Hospital, Cyprus. The target group consists of 176 patients of 32–89 years old. The dataset includes physical and biochemical examination results. Some patients took physical and biochemical examinations every 5–6 months, some others took examinations once a year and some took three examinations over the four years period.

The principal goal of the dataset collection was to identify the risk factors of coronary

artery disease and their impact on patients' health. Risk factors are defined as: smoking, diabetes and arterial hypertension considering systolic and diastolic blood pressure, high levels of cholesterol, high levels of triglycerides, low HDL, high LDL and overweight. The inclusion criteria to the study included the presence of at least one event, such as acute coronary syndrome (ACS), acute myocardial infarction (AMI), percutaneaous coronary intervention (PCI) and coronary artery bypass graft surgery (CABG).

2.2 Methodology

The basic steps of the proposed methodology are:

1. Data cleaning which includes identifying fields, selecting variables for abstraction and coding data as shown in Table 1.
2. State and trend abstraction techniques will be used to abstract concepts based on an interval duration given by a domain (12-months). State abstractions will be generated based on domain expert knowledge whereas trend abstractions will be generated by extending the algorithm described in [14].
3. The derived state abstractions will be the new concepts which represent the presence or absence of risk factors during the 12-month period for each patient.
4. A DBN model will be constructed using three time slices and each time slice will represent the time period of 12 months (e.g. 01/01/2009–01/01/2010). Nodes represent the abstract concepts which are binary variables and edges represent the dependencies between the concepts through the same time-slice or through two consecutive time slices. The structure of the DBN (prior and transition model) representing the state abstraction concepts is based on domain expert knowledge and medical literature as shown in Fig. 1.
5. Expectation Maximization learning algorithm [10] will be used for learning parameters of the model using the abstracted data set. Bayesian Net Toolbox for Matlab [11] will be used for the construction of the DBN. The constructed model will be able to make predictions for the risk of coronary artery disease for a specific patient.
6. The junction tree inference algorithm [11] will be used to compute the prediction risk (probability). If the probability will be over 0.5 then the risk of the presence of cardiovascular disease is severe otherwise it is normal.

Table 1 Variables for abstraction and coding data.

Variables	Code
Hypertension	HT
High Total Cholesterol	HTC
Low HDL	LHDL
High LDL	HLDL
High Triglycerides	HTR
High Glucose + Diabetes	DM
Smoking	SK
Overweight	BMI
Myocardial Infarction	MI
Acute Coronary Syndrome	ACS
Coronary Artery ByPass Graft	CABG
Percutaneous coronary intervention	PCI
Diet, Exercise	DIET, EX
Risk of CAD	hidden variable – CAD

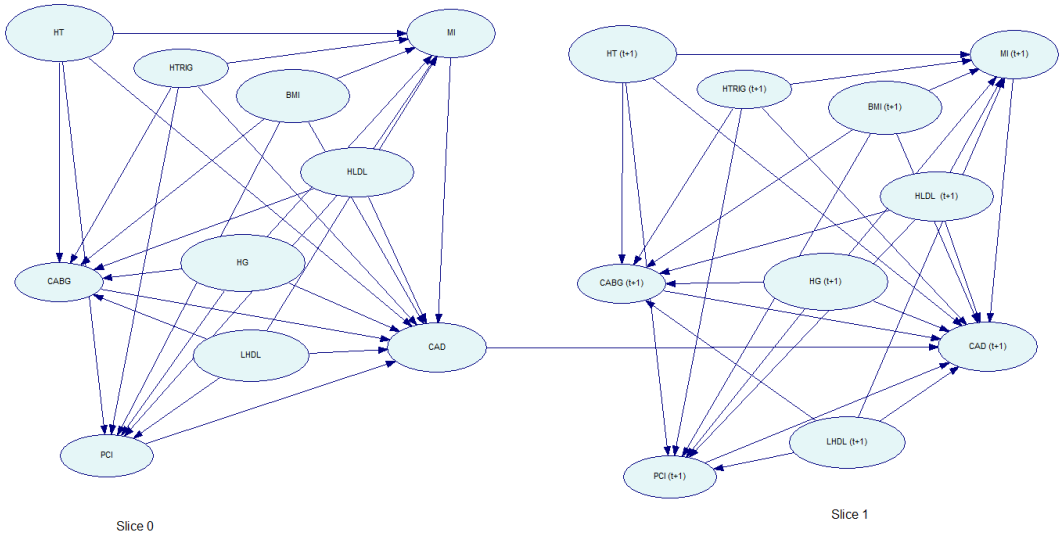


Figure 1 Coronary Artery Disease Model Structure Over Two Time Slices (t=0 and t=1): Nodes represent some of the variables as displayed in Table 1 and edges represent their dependencies (local and transitional relations). A transitional relation exists only with respect to the hidden node CAD.

2.3 Temporal Abstractions

State abstractions correspond to expressions like high blood pressure associated to a time interval in which such behavior occurs. The labels given to the associated intervals are: 'normal levels' (value = 1) or 'abnormal levels' (value=2) which are given by the domain expert as shown in Table 2. If a patient took more than one examination over the same year and the value of the state abstraction remains the same, the time points are joined into a maximal interval [Iss, Ies]. Alternatively, if a patient took more than one examination in the same year, but the state value of a variable is not the same for all examinations, then the following rules given by domain expert are applied:

- If a patient took two examinations during the desired time period and Vs = 2 (abnormal) at t1 but Vs = 1 (normal) at t2 then a risk factor is absent at [t1-t2] e.g patient with high cholesterol during his/her first examination and cholesterol value decreases to normal levels from the first examination to the next one, then the risk of hypercholesterolaemia is absent during that period. Similarly, if the cholesterol value increases from normal (at t1) to abnormal levels (at t2) then the risk of hypercholesterolaemia is present.
- If a patient took three examinations during the desired time period and the value of state abstraction is the same during t1 and t2 then the presence of a risk factor depends on the value of the abstraction at t1 and t2. For example, if Vs = 2 (abnormal) at t1 and t2 but Vs = 1 (normal) at t3 then a risk factor is present during the period [t1 – t3].
- If a patient took three examinations during the desired time period and the value of state abstraction is the same during t2 and t3 then the presence of a risk factor depends on the value of abstraction at t2 and t3. For example, if Vs = 1 (normal) at t1 but Vs = 2 (abnormal) at t2 and t3 the risk factor is present during the period [t1 – t3].
- If a patient took three examinations during the desired time period but the values of state abstractions are different at all three time steps then the presence of risk factors depends only on the value of abstraction at t3. For example if Vs = 2 (abnormal) at t1, Vs =1 (normal) at t2 and Vs = 2 (abnormal) at t3 then a risk factor is present during the period [t1 – t3].

- As concerned the risk factor of hypertension which depends both on systolic and diastolic blood pressure, the above rules have to concern either systolic or diastolic blood pressure values.

Trend abstractions will be generated after applying a median filter to the dataset for removing noise. Then, the values of variables occurring at consecutive time points will be compared. Let us assume that t1 is the time of the first consultation and t2 is the time of the second consultation during the year. If the value of a variable at t1 (V1) is equal to the value of the variable at t2 (V2) then the value of the variable V (Vt) during the interval [t1 t2] is 'steady'. If the value of a variable at t1 (V1) is less than the value of the variable at t2 (V2) then the value of the variable V (Vt) during the interval [t1 t2] is 'increasing', otherwise, the value of the variable V (Vt) during the interval [t1 t2] is 'decreasing'. A maximal interval where this behavior (state, increasing or decreasing) persists is derived.

Table 2 Categories of state abstractions as given by domain expert where each variable can take two possible state values depending on its raw value.

State Value	Normal for all patients	Normal for patients with Diabetes	Abnormal before the day of the event
Cholesterol (mg)	< 190	<170	
HDL (mg)	> 40	>40	
LDL (mg)	<100	<100	
Glucose (mg)	<110	<110	
BMI	<25	<25	
SBP (mmHg)	<130	<120	Hypertension = YES
DBP (mmHg)	<90	< 85	Hypertension = YES
Triglycerides (mg)	<150	<150	
Smoking	NO		EX (ex smoker)
Diet	YES		
Exercise	YES		

3 Related Work

Techniques/methods from these two areas are largely used independently of each other in many clinical domains and thus no specific integrations have been reported yet. Consequently, the goal of the proposed methodology is to combine temporal abstraction with Dynamic Bayesian networks and to apply this integration under the CAD clinical domain by developing a prediction model.

Several systems had been designed to abstract meaningful clinical concepts from raw clinical data such as TOPAZ, IDEFIX and VM, each one using its own abstraction methodologies [8, 5, 6]. The best known framework for temporal abstraction in clinical data, called Knowledge Based Temporal Abstraction (KBTA) was proposed by Shahar in [15] and implemented in many clinical domains. KBTA decomposes the temporal abstraction task into five computational subtasks, solved by corresponding temporal mechanisms. However, none of these approaches are able to deal with missing data and of the uncertainty that typically underlie clinical raw data. The KBTA method complete missing values only for bridging gaps between two intervals, in which the proposition (e.g.,anemia level) had the same value (e.g., moderate anemia). The IDEFIX system can deal with uncertainty and also with missing data using the time-of-validity slot which specifies for how long the value of the attribute is considered valid. Moreover, when domain medical knowledge is not enough and prior probabilities about some disease are missing, the derived conclusions of IDEFIX

may not be valid. Ramati and Shahar [12] proposed a new methodology called Probabilistic Temporal Abstraction (PTA) to perform a temporal abstraction task to clinical raw data using a probabilistic approach. This approach is able to eliminate uncertainty and to deal with missing values.

Considerable work on dynamic models in medicine has been carried out by Leong and collaborators who have successfully used a combination of graphic models with Markov chains to solve problems in different medical domains such as head injury management [7], colorectal cancer management, neurosurgical intensive care unit monitoring and palate management [18]. Other applications of Dynamic Bayesian Networks in medicine include forecasting sleep apnea [4], management of patients with carcinoid tumor [17] and diagnosis and decision making after monitoring patients suffering from renal failure and treated by hemodialysis [13].

A classification/diagnosis model under the clinical domain of CAD had also been developed using decision trees in a previous work by Karaolis [9] using a similar dataset. Preprocessing of current dataset is based on this work. Related is also the work in reasoning over multiple levels of temporal granularity through Bayesian networks [2]. In this work, two approaches were proposed to incorporate temporal abstractions and explicitly represent complex temporal relationships using fluents and hierarchical Bayesian networks.

4 Conclusion and Future Work

The perceived advantages of combining TA with DBN are that this integration can handle incomplete evidence and uncertainty estimating disease outcomes which are usual problems in clinical systems and also represent current limitations in datasets. DBN can provide a concrete understanding of how causal dependencies and temporal precedence between abstract concepts influence a particular disease outcome. Moreover, this integration facilitates the main advantage of BNs which is their capability to integrate expert's knowledge with empirical data to model a disease.

Another advantage of integrating a DBN with TA is the ability to represent high-level abstracted concepts rather than low-level data, thus making the models simpler and at the same time more conceptual. Moreover, the interpretation of the prediction results will be context-based, allowing for more accurate decision making. In addition, uncertainty and errors in clinical datasets are very common and TA and DBN techniques can deal with such error-prone measurements.

In this paper, a preliminary approach of integrating temporal abstraction with DBN in CAD clinical domain is presented. The proposed approach has some limitations which will be overcoming at a later stage of this study. Although the dataset includes information about history of hypertension, diabetes, MI and ACS which may have occurred before 2009, the constructed model can only represent risk factors and events that hold during the period 2009–2012. The model should be extended in order to represent history of events as well. Another possible improvement is the developed model to be able to facilitate continuous improvement and innovation in medical tasks through incremental learning. This means that as new cases will be dynamically added to the network, the model must self-adapt its structure and adjust its parameters on-line. Furthermore, later stages of this study will work on how trend abstractions can be represented in a DBN and the possibility to develop an hierarchical model with abstracted data of different granularities.

References

1 Larizza C. Bellazzi R. and Riva A. Temporal abstractions for interpreting diabetic patients monitoring data. *Intelligent Data Analysis*, 2(1-4):97–122, 1998.

2 Brendan Burns and Clayton T. Morrison. Temporal abstraction in bayesian networks. In *AAAI Spring Symposium*, 2003.

3 T. Charitos. *Reasoning with dynamic networks in practice*. PhD thesis, Utrecht University, Netherlands, 2007.

4 Paul Dagum and Adam Galper. Forecasting sleep apnea with dynamic network models. In *Proceedings of the Proceedings of the Ninth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-93)*, pages 64–71, San Francisco, CA, 1993. Morgan Kaufmann.

5 Isabelle de Zegher-Geets. Idefix: Intelligent summarization of a time-oriented medical database. Technical Report KSL-88-34, Knowledge Systems, AI Laboratory, 1987.

6 L.M.I. Fugan. *VM2 representing time-dependent relations in a medical setting*. PhD thesis, Stanford University, 1980.

7 D. Harmanec, TY Leong, S. Sundaresh, KL Poh, TT Yeo, I. Ng, and TW Lew. Decision analytic approach to severe head injury management. In *Proceedings of the AMIA Symposium*, page 271. American Medical Informatics Association, 1999.

8 L. B. Sheiner M. G. Kahn, L. M. Fagan. Combining physiologic models and symbolic methods to interpret time-varying patient data. *Methods of Information in Medicine*, 30(3):167–168, August 1991.

9 D. Hadjipanayi M.A. Karaolis, J.A. Moutiris and C.S. Pattichis. Assessment of the risk factors of coronary heart events based on data mining with decision trees. *Information Technology in Biomedicine, IEEE Transactions on*, 14(3):559–566, 2010.

10 T.K. Moon. The expectation-maximization algorithm. *Signal Processing Magazine, IEEE*, 13(6):47–60, 1996.

11 K. Murphy et al. The bayes net toolbox for matlab. *Computing science and statistics*, 33(2):1024–1034, 2001.

12 Michael Ramati and Yuval Shahar. Probabilistic abstraction of multiple longitudinal electronic medical records. In *Proceedings of the 10th conference on Artificial Intelligence in Medicine*, AIME'05, pages 43–47, Berlin, Heidelberg, 2005. Springer-Verlag.

13 C. Rose, C. Smaili, and F. Charpillet. A dynamic bayesian network for handling uncertainty in a decision support system adapted to the monitoring of patients treated by hemodialysis. In *Tools with Artificial Intelligence, 2005. ICTAI 05. 17th IEEE International Conference on*, pages 5–pp. IEEE, 2005.

14 A. Salatian and J. Hunter. Deriving trends in historical and real-time continuously sampled medical data. *Journal of intelligent information systems*, 13(1):47–71, 1999.

15 Y. Shahar and M.A. Musen. Knowledge-based temporal abstraction in clinical domains. *Artificial intelligence in medicine*, 8(3):267–298, 1996.

16 M. Stacey and C. McGregor. Temporal abstraction in intelligent clinical data analysis: A survey. *Artificial Intelligence in Medicine*, 39(1):1–24, 2007.

17 M.A.J. Van Gerven, B.G. Taal, and P.J.F. Lucas. Dynamic bayesian networks as prognostic models for clinical patient management. *Journal of biomedical informatics*, 41(4):515–529, 2008.

18 Yanping Xiang and Kim-Leng Poh. Time-critical dynamic decision making. In *UAI*, pages 688–695, 1999.

Get started imminently: Using tutorials to accelerate learning in automated static analysis

Jan-Peter Ostberg and Stefan Wagner

University of Stuttgart, Institute of Software Engineering
Universitätstr. 38, 70569 Stuttgart, Germany
{jan-peter.ostberg},{stefan.wagner}@informatik.uni-stuttgart.de

Abstract

Static analysis can be a valuable quality assurance technique as it can find problems by analysing the source code of a system without executing it. Getting used to a static analysis tool, however, can easily take several hours or even days. In particular, understanding the warnings issued by the tool and rooting out the false positives is time consuming. This lowers the benefits of static analysis and demotivates developers in using it.

Games solve this problem by offering a tutorial. Those tutorials are integrated in the setting of the game and teach the basic mechanics of the game. Often it is possible to repeat or pick topics of interest. We transfer this pattern to static analysis lowering the initial barrier of using it as well as getting an understanding of software quality spread out to more people.

In this paper we propose a research strategy starting with a piloting period in which we will gather information about the questions static analysis users have as well as hone our answers to these questions. These results will be integrated into the prototype. We will evaluate our work then by comparing the fix times of user using the original tool versus our tool.

1998 ACM Subject Classification D.2.5 Testing and Debugging, D.2.9 Management, H.5.2 User Interfaces

Keywords and phrases static analysis, motivation, usability, empirical research, gamification

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.109

1 Introduction

No one wants to read an encyclopaedia to play a game. –Nolan Bushnell

Many companies are aware of the potential benefits of static analysis, such as increased maintainability of source code[1, 8], but they are afraid to invest the time which a developer needs to get used to a static analysis tool. To a certain degree this fear is justified, because it could take hours to days to understand the basics and many years to fully grasp the warnings and mechanics of the static analysis as well as the implications to software quality. Additionally, it is hard to justify these spendings, if the results are not delivered in a short time and if there is little to no knowledge about software quality and especially maintainability.

Gamers today are in some way similar to these companies. They also do not want to invest much time and effort before they can start experiencing their virtual worlds and have fun. Knowing this, the game developers often have a tutorial stage included into their games, which walks the players through the basic concepts of the game. This tutorial is linked to the game's story and thereby also functions as an introduction to the context, the so called *setting*. Also, to not annoy the experienced or returning gamer, these tutorials are not mandatory to play, but can be re-entered any time later, if the player feels the need to.

So why not bring this idea to static analysis? Besides the reduction of the time to get used to a static analysis tool, the introduction to the context of software quality could lead

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 109–115

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

more people to a deeper understanding of what software quality is and how it is influenced. This also may enable the users of the tutorial to argument more efficiently with their newly acquired knowledge in favour of the benefits of static analysis. In addition, expert knowledge can be used to customise the tools, which leads to a significantly reduced rate of false-positives as shown by Wagner et al. [13]. In the following, we will lay out a research strategy to cover the current state of the problem, create a suitable prototype and evaluate the benefit of the idea.

2 Related Work

The work by Zheng et al. [14] describes the areas which can be improved by static analysis. In conclusion, the authors state that the use of static analysis could eliminate the less sophisticated faults, freeing up time and capacities to work on the faults which need more thorough analysis by human beings. We can conclude that there is a significant benefit in knowing how to use static analysis and an in-depth knowledge can reduce the amount of time for the analysis, freeing up even more time.

Ayewah and Pugh [2] look into how the static analysis tool FindBugs is used in companies. In detail, they are interested in how they conquer the initial hump due to the high number of warnings at a first time use, as well as the processes used to keep warnings from reappearing. The results show that developers are interested in almost every warning. Also the authors are able to identify some of the processes used by companies to make their work with static analysis more effective.

A view from the creator's side is shown by Bessey et al. [3]. They describe what problems they faced, when they introduced their scientific tool to the "real world". This is a nice example of what is possible today and what it needs to get a tool into everyday practice, for example a well designed installation process or less sophisticated, but understandable analysis.

Pagulayan et al. [10] talk about stumble stones in game development. They point out that if a system is complex, it will profit from a tutorial. They also point out, however, that a tutorial has to follow certain rules, like finding the right pace or not bore the player with tedious instructions.

James Paul Gee [4] has detailed comments about what a good tutorial should provide. He shows this on examples of prominent games. He also points out the possible negative results a bad designed tutorial can have. Both of these sources show that a good tutorial will help get the player more deeply involved, but we have to value some rules, when we create our tutorial, to not generate a negative effect with it. These rules are, for example, a well set pace of difficulty increase as well as not to force the user to strictly go through the whole tutorial.

Randel et al. [11] conducted a literature study on the question of the effectiveness of games for educational purposes. The authors concluded, that depending on a number of variables, e.g. cognitive learning style, games can help learning, because they demand interaction of the player. This interaction increases the chances of the material to be integrated into the cognitive memory and so be remembered more easily. This implicates for our idea that with the tutorial idea, which is close to games, we might have a higher chance of having the users remembering the tool usage.

3 Research Goal

Static analysis tools should be used more commonly in software development, because this would increase the overall quality of the software created. With our research, we aim to find the reasons why they are not widely used and develop strategies to address these problems. In this paper, we focus on a way to shorten the time needed for understanding how to use automatic static analysis and the time needed to understand its analysis results. We think understanding static analysis is a problem, because we observed that in many companies, which use static analysis tools, there is only one person, who takes care of the tool, making the configurations and maintaining the process. Most of the other employees have little to no knowledge of the tool besides starting it. This shows that companies do not want each of their employees to invest the same amount of time into getting familiar with the tool, because they are aware that it is time intensive.

To address this problem, we build on ideas from games which is also known as gamification ([12], [6]). Built-in tutorials, which explain the basic mechanics step by step, seem to us as promising. In the following we well lay down in detail a research strategy to research this hypothesis.

4 Example Static Analysis Tool: FindBugs

There is a huge amount of tools for automatic static analysis available. The abilities of these tools vary from simple style checking to highly sophisticated analyses. Also the form of licensing ranges from free open source to very expensive pay-per-use models. From this motley crew of tools we decided to take FindBugs[1]. FindBugs is a open source tool, distributed under the terms of the Lesser GNU Public License and was developed at the university of Maryland. It uses rather simple rules[2] to find problematic part in Java byte code [1]. This and the fact that it is free of charge makes it one of the most popular analysis tools. By deciding to modify this tool we will have no licensing problems and can benefit from a large community, which we later can offer our modification and so hopefully get feedback for further improvement.

5 Research Strategy

In the following we take a look at the steps we aim to take for our research, which will be described in more detail in the following subsections. The first step will be piloting our idea to gain a feeling for the problems of the users. Step number two will be the creation of a prototype utilizing the information of the pilot. With the prototype done, we can start, as step three, to evaluate the impact of our ideas. To reduce the amount of data to a manageable size, we will first focus on the static analysis tool FindBugs. We decided on this tool, because due to its open source nature it is easily accessible and extendible. We will also provide a code base for the experiments on which the analyses are conducted on.

5.1 Step 1: Piloting the Idea with a Tutor

Before we create a first prototype of the built-in tutorial, we will assess the possible benefit of the tutorial with a tutor in person. The "Wizard of Oz" [5, 7] research method is the

[1] http://findbugs.sourceforge.net/
[2] http://findbugs.sourceforge.net/bugDescriptions.html

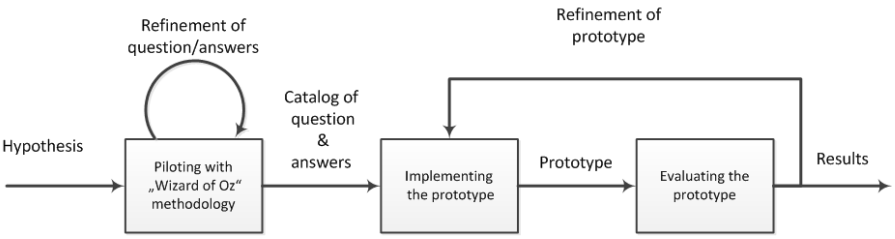


Figure 1 Schematic of the research strategy.

method of choice here. The "Wizard of Oz" is a research method where the participant of the experiment is not aware that he or she is interacting with a human being which is simulating the intelligent tool. We will be able to easily adapt to questions and problems the user has and so have a close feedback loop to refine our question/answer catalogue. We can achieve this in our experiment by using a screen sharing tool like Skype or Teamviewer and a chat. The screen sharing will be hidden to the participant and the chats optic will be modified so that the participants cannot recognise it as such. By conducting this pilot, we will be able to gather information on what the users are interested in to learn from a tutorial. We will be able to adapt our ideas in the piloting phase until they fit the users demand more closely and so deliver a more satisfying experience. The participants of the pilot will be recruited from the students of our university with various study courses to represent the various stages of IT knowledge in the real world. We plan to perform the experiment only with three to 10 students because this setting is time-intensive and, as the sole purpose of the pilot is to assess the feasibility and acceptance of the idea, we expect to gather enough information for the next step with these numbers, as Nielsen et al. [9] shows that 5 testers will find about 85% of the problems. Additionally, we have access to information gathered by a recent study conducted by us, were we observed first time users of FindBugs with an eye tracker and think aloud. These results will also have an influence to our prototype.

5.2 Step 2: The Prototype

With the gathered data from the pilot phase, we will be able to create a first built-in tutorial prototype. For the reward system, we would like to provide, we take some inspiration from ribbon hero[3] created by the Microsoft Office labs. The first challenge to overcome here is finding a story to tell which is related to the topic, not too plain and not too complicated or weird. The setting of a detective story seems to be a nice fit, as we can handle categories of findings as ”cases”. The second challenge will be the design of an engaging experience points system which is fair and comprehensible to the users. The built-in tutorial will offer tasks which wili correspond to the fix of warnings issued by the analysis tool. The tasks issued by the tutorial will increase in difficulty but should never ask too much of the user. Also the user should be able to skip most of the tutorial but has to demonstrate his or her understanding of the task by completing it.

The task should be taken from the source code he or she wants to analyse, but we will have ”back-up” code for the tasks that are not contained in the provided code but still are necessary to be learned for the optimal usage of the tool. This ”back-up” code could also be used as an additional example for tasks that are hard to understand with the code provided

[3] `http://www.ribbonhero.com/`

or if the user requests another example. To get the user more engaged, we will reward the solution of tasks with some kind of point system to make the increase in knowledge visible for the user.

To make the benefit of the "back-up" code more clear, let us consider the following example. We have analysed the source code of JabRef[4], an open source reference manager. One warning issued by Findbugs is:
"... /SampleCode (JabRef)/... /jabref/imports/PdfXmpImporter.java:48
VERY confusing to have methods net.sf.jabref.imports.PdfXmpImporter.getCLIid() and net.sf.jabref.imports.ImportFormat.getCLIId()".
This warning is rather cryptic and hard to understand. Even if you take a look at the code, it might take a long time before you realise what is the problem addressed here. The user might want to have a more easy example of source code which would provoke the same type of warning. An early mock-up of the tutorial information is presented in figure 2. With this

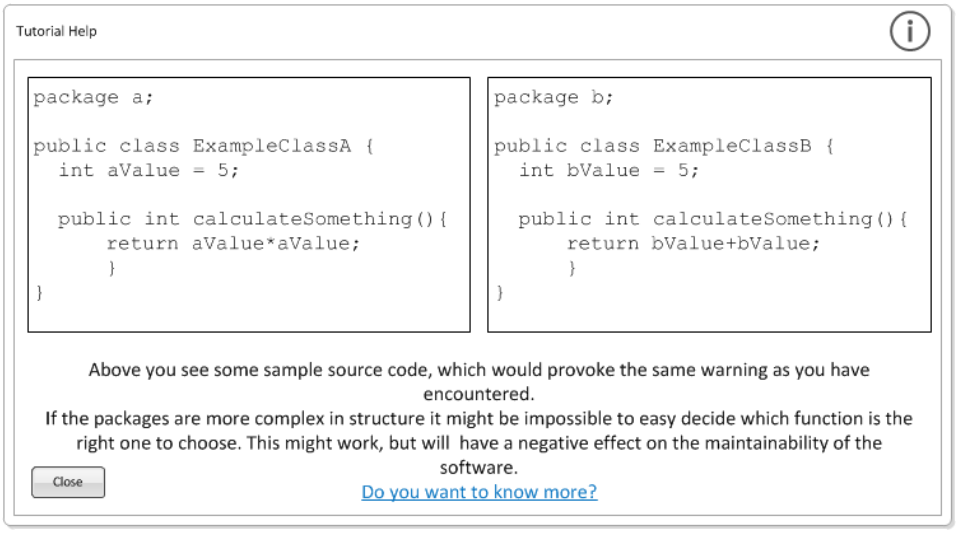


Figure 2 Mockup of prototype tutorial window.

example it is clearer that the warning issued should remind the creator of the code that it is not a good idea to have two methods with the same name in the same project doing different things. As we still follow the tutorial idea, we would also offer some information text, which will provide a link to even more detailed information comparable to an encyclopaedia, for those user, who really want to master the tool and want to dive deep into the ideas of software quality. This should make the proposed changes of the tool and the tutorial more convincing to the users as well as help them to understand how to create software of higher quality and avoid future mistakes.

5.3 Step 3: Evaluation of the Prototype

After the prototype is finished, we are planning an evaluation phase. To evaluate the benefit of our approach we will compare the time needed to fix a given set of warnings in a code base of participants using our enhanced tool versus participants using the original tool. We will

[4] `http://jabref.sourceforge.net/`

take the time from start to finish of the whole operation as well as of the fix time only. To do that, we will measure how long the participants work with the tutorial and how much time the other participants invest into getting to know the tool. We expect that especially the more complex problems will be solved faster by user with our tutorial approach. Additionally we ask the participants afterwards to use the tool, they did not use yet. Here the time is, of course, not measured, because the participants already learned from the other tool. We will issue questionnaires then to cover the subjective helpfulness of the tutorial for getting started with FindBugs and static analysis, as well as the level of engagement created through the gamification of the tool versus the unmodified tool. For example, planed question are:

- Do you think the enhanced tool is faster understandable then the original one?
- Do you feel more motivated to fix issues by the game mechanics?
- Have you used the possibility to gain more information considering software quality?
- Did the tool raise your interest in software quality?
- Would you prefer to use the enhanced or the original tool?
- ...

To make the answers more comparable, we will offer four possible answers for the question that do not need a more complex answer.

- "Yes, I fully agree."
- "Yes, I agree mostly."
- "No, I do not agree to that."
- "No, I completely disagree with that."

Moreover, we will have a section where the participant can give free feedback on the prototype. We will carefully keep track of this feedback and use it from time to time to make useful improvements to the software. For a long term evaluation, we consider to include the prototype as a lecture accompanying instrument in teaching. Here we plan to use our tool for a whole semester and ask the students at the end of the semester to state their experience with the tool. The details of this are subjects of future work.

6 Summary and Future Work

We presented our overall research goal, which is making automatic static analysis a more common tool in software development. In this paper we propose a tutorial attempt to shorten the time needed to get started with static analysis. As a side effect the proposed idea will teach the willing user to learn the ideas behind the issued warnings. These ideas reach into software quality and software engineering topics. We laid out a research strategy to create a problem oriented catalogue of questions and answers for our tutorial by a pilot study and to evaluate the benefits of our approach.

There are other aspects of the automated static analysis, that might make it unattractive and is not covered here. For example the problems could originate from a poor operability. We are planning to examine this aspect with an eye tracking study which we will conduct shortly. Finally, there is still the problem with the false positives. Future work will also aim to find techniques to reduce or make them easier to spot and track.

References

1 N. Ayewah, D. Hovemeyer, J.D. Morgenthaler, J. Penix, and W. Pugh. Using static analysis to find bugs. *IEEE Software*, 25(5):22–29, 2008.

2 Nathaniel Ayewah and William Pugh. A report on a survey and study of static analysis users. In *Proceedings of the 2008 workshop on Defects in large software systems*, DEFECTS '08, pages 1–5. ACM, 2008.

3 Al Bessey, Ken Block, Ben Chelf, Andy Chou, Bryan Fulton, Seth Hallem, Charles Henri-Gros, Asya Kamsky, Scott McPeak, and Dawson Engler. A few billion lines of code later: using static analysis to find bugs in the real world. *Communications of the ACM*, 53:66–75, 2010.

4 James Paul Gee. What video games have to teach us about learning and literacy. *Comput. Entertain.*, 1(1):20–20, October 2003.

5 J. F. Kelley. An iterative design methodology for user-friendly natural language office information applications. *ACM Trans. Inf. Syst.*, 2(1):26–41, January 1984.

6 Jane McGonigal. *Reality Is Broken: Why Games Make Us Better and How They Can Change the World*. The Penguin Group, 2011.

7 Lennart Molin. Wizard-of-oz prototyping for co-operative interaction design of graphical user interfaces. In *Proceedings of the third Nordic conference on Human-computer interaction*, NordiCHI '04, pages 425–428, New York, NY, USA, 2004. ACM.

8 Nachiappan Nagappan and Thomas Ball. Static analysis tools as early indicators of pre-release defect density. In *Proceedings of the 27th international conference on Software engineering*, ICSE '05, pages 580–586. ACM, 2005.

9 Jakob Nielsen and Thomas K. Landauer. A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, CHI '93, pages 206–213, New York, NY, USA, 1993. ACM.

10 Randy J. Pagulayan, Kevin Keeker, Dennis Wixon, Ramon L. Romero, and Thomas Fuller. The human-computer interaction handbook. chapter User-centered design in games, pages 883–906. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 2003.

11 Randel. The Effectiveness of Games for Educational Purposes: A Review of Recent Research. *Simulation Gaming*, 23:261–276, 1992.

12 Byron Reeves and J Leighton Read. *Total Engagement: Using Games and Virtual Worlds to Change the Way People Work and Businesses Compete*. Harvard Business School Press, 2009.

13 S. Wagner, F. Deissenboeck, M. Aichner, J. Wimmer, and M. Schwalb. An evaluation of two bug pattern tools for Java. In *Proc. 1st International Conference on Software Testing, Verification, and Validation (ICST'08)*, pages 248–257. IEEE Computer Society, 2008.

14 J. Zheng, L. Williams, N. Nagappan, W. Snipes, J.P. Hudepohl, and M.A. Vouk. On the value of static analysis for fault detection in software. *IEEE Transactions on Software Engineering*, 32(4):240–253, 2006.

A Quantitative Study of Social Organisation in Open Source Software Communities*

Marcelo Serrano Zanetti, Emre Sarigöl, Ingo Scholtes, Claudio Juan Tessone, and Frank Schweitzer

Chair of Systems Design, ETH Zurich, Switzerland
{mzanetti,semre,ischoltes,tessonecfschweitzer}@ethz.ch

Abstract

The success of open source projects crucially depends on the voluntary contributions of a sufficiently large community of users. Apart from the mere size of the community, interesting questions arise when looking at the *evolution of structural features* of collaborations between community members. In this article, we discuss several network analytic proxies that can be used to quantify different aspects of the social organisation in social collaboration networks. We particularly focus on measures that can be related to the cohesiveness of the communities, the distribution of responsibilities and the resilience against turnover of community members. We present a comparative analysis on a large-scale dataset that covers the full history of collaborations between users of 14 major open source software communities. Our analysis covers both aggregate and time-evolving measures and highlights differences in the social organisation across communities. We argue that our results are a promising step towards the definition of suitable, potentially multi-dimensional, resilience and risk indicators for open source software communities.

1998 ACM Subject Classification D.2.8 Metrics, K.4.3 Organisational Impacts

Keywords and phrases open source software, mining software repositories, social networks

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.116

1 Introduction

What are the most important social factors that lead to successful and sustainable open source software projects? According to *Linus' Law* - which states that "given enough eyeballs, all bugs are shallow" [7] - the quality and success of open source software (OSS) critically depends on the existence of a sufficiently large community of developers who review, modify and improve the publicly available source code. Apart from development efforts, another important success factor is the existence of a stable community of users who report software defects, request and inspire new features, reproduce bugs or comment on issues reported by other users. By employing the collective knowledge and diverse experiences of many contributors, most OSS communities manage to provide technical assistance to less experienced users, often on a time scale that is competitive to commercial software support.

Depending on the distribution of competencies and responsibilities of contributors, largely different patterns of collaborations may arise. While it is generally difficult to assess these social factors of OSS projects, the availability of large scale data on community dynamics increasingly allows to study the *social dimension of OSS projects* from a quantitative perspective [8, 16]. Previous studies have mainly focused on rather simple proxies of social

* This work was supported by the SNF through grant CR12I1_125298. C.J Tessone acknowledges financial support from the SBF through grant C09.0055.

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 116–122

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

dynamics like the evolution of the number of contributors and contributions or the time span of a user's activity and were mostly based on a rather limited set of snapshots of a single project. Using a large scale dataset of time-stamped social interactions that has been collected from the Bugzilla bug-tracker installations of 14 major OSS projects, in this paper we study the *fine-grained evolution of structural features of networks of user collaborations*. We thus take a *network perspective on OSS communities* and highlight differences in the social organisation of software projects that can be related to their activity, their cohesion as well as their resilience against fluctuations in the community members. By applying standard measures from social network analysis we particularly quantify how tightly community members collaborate, how equal responsibilities are distributed and how resilient collaboration topologies are against the loss of (central) community members. While similar tools have been applied to OSS projects before [3, 6], to the best of our knowledge, the present paper is the first to study these network analytic measures on a dataset that covers the full, fine-grained history of 14 well-established and successful OSS communities.

2 Social Organisation in OSS Communities: A Network Perspective

In order to make substantiated statements about the structure and dynamics of the social organisation of OSS communities, we recently completed collecting data on the history of user collaborations recorded by the Bugzilla installation of 14 well-established OSS projects. Bugzilla[9] is an open source bug tracking system which is utilised by users and developers alike to report bugs, keep track of open issues and feature requests and comment on issues reported by others. Since the Bugzilla installations of OSS projects are used to foster collaboration between community members, it constitutes a valuable source of data that allows us to track social interactions between developers and users.

2.1 Building Social Networks from Bug-Reports

Data in the Bugzilla database are arranged around the notion of *bug reports*. Each bug report has a set of fields describing aspects like the user who initially filed the bug report, its current status (e.g. *pending*, *reproduced*, *solved*, etc), to whom the responsibility to provide a fix has been assigned, attachments which may be used to reproduce or resolve the issue, comments and hints by other community members, or a list of community members which shall be informed about future updates. Apart from an initial bug report, Bugzilla additionally stores the full history of all updates to any of the fields of a bug report. Each of these change records includes a time stamp, the ID of the user performing the change as well as the new values of the changed fields. While our dataset comprises change records for all possible fields, in this article we focus on those that indicate changes in the users that are assigned responsibility to fix an issue (henceforth called the *ASSIGNEE* field) and changes to the list of users to whom future updates of the bug shall be sent via E-Mail (henceforth called the *CC* field). We consider any updates in the *CC* and *ASSIGNEE* field of a bug report as a time-stamped edge from the user who performed the update to the user(s) who were added to the *CC* field or the *ASSIGNEE* list of responsible developers respectively.

Based on the data extraction procedure described above, we obtain a large time-aggregated network of nodes representing community members and time-stamped edges representing a particular interaction between two users. For most of the projects considered, the Bugzilla history from which we extract the network is longer than ten years. The fact that - in social networks aggregated over such long periods of time - most of the users represented by nodes have never been active within the same time period limits the expressiveness of the network

structure in terms of a project's "social organisation". In order to overcome this issue, we perform a *dynamic network analysis* by defining a sequence of *monthly collaboration networks* based on the time stamps of edges. In particular, we define a 30 day sliding time window and filter out those edges whose time stamps are outside the window and those nodes who did not have any interactions in the corresponding time period. By progressively advancing the start date of the sliding 30 day time window by one day increments we obtain a sequence of collaboration networks that allows us to study the structure of the community's social organisation as well as its evolution over time. Naturally, most of the monthly networks obtained in the way described above will not be fully connected. Since the network analytic measures we intend to apply assume connected topologies, we perform a component analysis on all snapshots and restrict our quantitative analysis to the largest connected component (LCC). In order to test the significance of our findings we further compute the fraction of those nodes who are part of the largest connected component. Table 1 shows the 14 OSS projects that are included in our dataset along with the time period and the total number of bug reports and updates that we included in our analysis. The column *LCC/TOTAL* furthermore indicates the fraction of users in the LCC, averaged over all monthly snapshots of the corresponding project. Here one observes that our data shows a rather large degree of variation with respect to this fraction, which may be seen as an argument that this measure is an interesting indicator for the *cohesiveness* of OSS communities by itself. Nevertheless, we argue that for all projects the fraction of users in the LCC is sufficiently large to make substantiated statements about the project's social organisation.

Table 1 Aggregated measures for the studied projects. From column *LCC/Total* to the last on the right, the numbers indicate the mean value ± standard deviation.

Project Name	Bugs	Updates	Period	LCC/Total	Nodes in LCC	Edges	Mean Degree	Assortativity	Closeness Central.	Clustering Coefficient
XAMARIN	4552	20721	2011-2012	0.93±0.05	46.76±8.12	98.15±22.70	2.07±0.29	-0.14±0.11	0.40±0.07	0.22±0.05
THUNDERBIRD	35388	313957	2000-2012	0.53±0.26	64.82±53.49	86.44±80.05	1.05±0.42	-0.23±0.17	0.40±0.27	0.04±0.05
LIBREOFFICE	8916	78341	2010-2012	0.78±0.11	73.83±32.06	114.41±49.10	1.56±0.26	-0.20±0.10	0.40±0.09	0.13±0.06
MAGEIA	6600	46921	2006-2012	0.93±0.07	77.54±21.80	156.00±59.24	1.95±0.30	-0.37±0.12	0.54±0.09	0.14±0.04
MANDRIVA	60546	368463	2002-2012	0.70±0.18	88.15±60.70	142.16±118.44	1.41±0.38	-0.29±0.15	0.40±0.14	0.07±0.05
FIREFOX	112953	1067914	1999-2012	0.58±0.23	171.77±117.79	240.79±180.44	1.16±0.44	-0.15±0.11	0.32±0.23	0.04±0.04
SEAMONKEY	90040	993392	1998-2012	0.67±0.15	210.39±251.43	364.42±482.54	1.48±0.48	-0.19±0.13	0.34±0.11	0.08±0.06
NETBEANS	210921	1875878	2000-2012	0.96±0.05	269.71±292.07	1069.72±1509.12	3.39±1.13	-0.12±0.08	0.37±0.05	0.23±0.08
OPENOFFICE	118135	915749	2000-2012	0.88±0.19	319.01±169.88	931.35±591.80	2.52±0.84	-0.12±0.10	0.34±0.15	0.12±0.06
GENTOO	140216	661783	2002-2012	0.80±0.07	338.97±110.86	617.73±211.92	1.82±0.27	-0.29±0.10	0.49±0.13	0.04±0.03
KDE	179470	648331	2002-2012	0.75±0.12	361.16±246.16	424.61±301.20	1.15±0.07	-0.16±0.07	0.32±0.07	0.01±0.01
ECLIPSE	356415	2594385	2001-2012	0.78±0.08	472.58±180.71	964.47±411.94	2.06±0.38	0.05±0.08	0.25±0.05	0.13±0.03
GNOME	550722	2751441	2000-2012	0.67±0.12	523.76±585.26	610.16±616.81	1.25±0.22	-0.17±0.09	0.25±0.08	0.03±0.04
REDHAT	414163	3777634	2006-2012	0.45±0.26	658.06±865.97	983.58±1297.18	1.19±0.35	-0.12±0.20	0.30±0.23	0.00±0.01

2.2 Network Measures

While the literature is rich in terms of measures able to quantify structural features of networks [11, 5], due to space limitations here we focus on three measures which are able to capture basic network qualities that relate to the *cohesiveness* of a community, the distribution of responsibilities among its members and its resilience against fluctuations in the user base. The first network measure is based on the *closeness centrality* of a node, which is defined as the inverse of the sum of the shortest path length to all other nodes in the network.

$$Cc(n_i) = \sum_{j=1, j\neq i}^{N} \frac{N-1}{d(n_i, n_j)} \in [0, 1] \tag{1}$$

where $Cc(n_i)$ corresponds to the *closeness centrality* score of node n_i, $d(n_i, n_j)$ is the length of the shortest path between nodes n_i and n_j, while N corresponds to the total number of nodes in a given network. Finally, the factor $N-1$ is a normalisation constant [2]. Based on this, the *closeness centralisation* of a network (Cc_{global}) can be calculated by taking the sum of the differences between the node with the highest value of closeness centrality (n^*) and the closeness centrality scores of all other nodes. This quantity is then normalised to the range of 0 to 1 using the theoretical value that results from a (maximally centralised) star network. Equation (2) presents the formal definition, while more details can be found in [2, 11]. In the context of OSS collaboration networks, closeness centralisation captures to what degree responsibilities, collaboration and communication are distributed equally across community members.

$$Cc_{global} = \sum_{i=1}^{N} \frac{Cc(n^*) - Cc(n_i)}{\frac{(N-2)(N-1)}{2N-3}} \in [0,1] \tag{2}$$

The second measure, the *clustering coefficient* of a network (C), measures how closely community members interact with each other in the sense that an interaction between a user X and Y, as well as an interaction between user Y and Z will also entail a direct interaction between the users Y and Z. The formal definition is presented in equations (3) and (4).

$$C(n_i) = \frac{2L_{D_{n_i}}}{D_{n_i}(D_{n_i}-1)} \in [0,1] \tag{3}$$

$$C = \frac{1}{N}\sum_{i=1}^{N} C(n_i) \in [0,1] \tag{4}$$

where D_{n_i} is the number of nodes directly connected to the node n_i, while $L_{D_{n_i}}$ is the number of edges between them. Therefore, the clustering coefficient $C(n_i)$ of node n_i expresses the fraction of edges that were realised from the possible $\frac{D_{n_i}(D_{n_i}-1)}{2}$ edges which are expected in a fully connected network with D_{n_i} nodes. We obtain the clustering coefficient of a network by averaging the clustering coefficient scores of all existing nodes (see equation (4)). This procedure can be seen as measuring how *cohesive* the community is in terms of nodes being embedded in collaborating clusters [11].

Finally, the *assortativity* (r) measures an individual's preference to connect to other individuals that have a similar or different degree of connectivity (the degree being a node's number of connections to different nodes). Networks in which nodes are preferentially connected to nodes with similar degree are called assortative. In this case a positive degree assortativity ($0 \ll r \leq 1$) indicates a positive correlation between the degrees of neighbouring nodes. Networks in which nodes are preferentially connected to nodes with different degree are called disassortative and in this case degree assortativity is negative ($0 \gg r \geq -1$). In networks with zero degree assortativity, there is no correlation between the degrees of connected nodes, i.e. nodes do not exhibit a preference for one or the other. Formally,

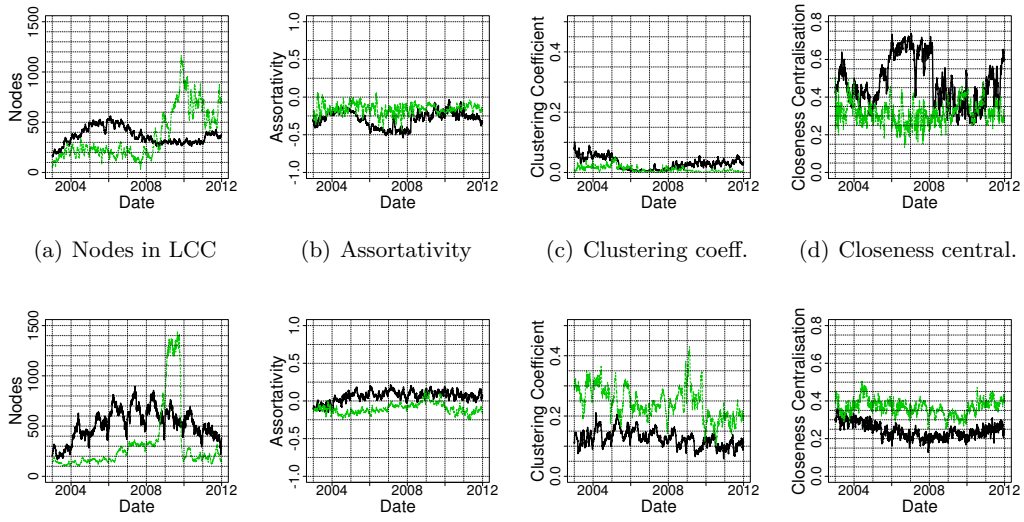
$$r = \frac{\sum_{ij} ij(e_{i,j} - q(i)q(j))}{\sigma(q)^2} \in [-1,1] \tag{5}$$

where e_{ij} is the fraction of all links in the network that join together nodes with degrees i and j, $q(i) = \sum_j e_{i,j}$, $q(j) = \sum_i e_{i,j}$ and $\sigma(q)$ is the standard deviation of the distribution of q. The term $q(i)q(j)$ is the equivalent to the expected value of $e_{i,j}$ inferred from a random network. Therefore, if $r = 0$ the pattern of interconnection between nodes is also random [4].

3 Comparative Analysis of OSS Communities

As described above, the preliminary results presented here have been obtained for the LCC of the network of monthly collaborations in terms of *CC* and *ASSIGNEE* interactions. While Table 1 shows the aggregate measures averaged over all time windows for every project in our database, due to space constraints we limit the presentation of the dynamics of the social organisation to the projects Gentoo and KDE (both Gnu/Linux related projects) as well as Eclipse and NetBeans (both Java IDEs). These have been chosen because a) their communities are of comparable size and age, b) the respective pairs of projects address similar problem domains and c) they represent contrasting examples with respect to the measures studied in this paper.

Figure 1 shows the evolution of the number of nodes in the LCC, its assortativity, clustering coefficient and closeness centralisation for these four projects. For all projects, the fraction of nodes in the LCC is rather stable with values between 0.7 and 1 consistent with the aggregate values given in Table 1. The same is true for the evolution of the mean degree. We thus omit these plots. The four projects show significant differences in the evolution of the clustering coefficient that cannot be explained by mere size effects. In the particular time frame between 2006 and 2008, the clustering coefficient of the Eclipse community (≈ 0.15) was roughly ten times higher than that of the Gentoo community (≈ 0.01), although the LCCs of both communities were of comparable size (≈ 500 nodes). In addition, the clustering coefficient of the Gentoo community shows an interesting dynamics, dropping to a very small value between 2006 and 2008 and increasing thereafter.

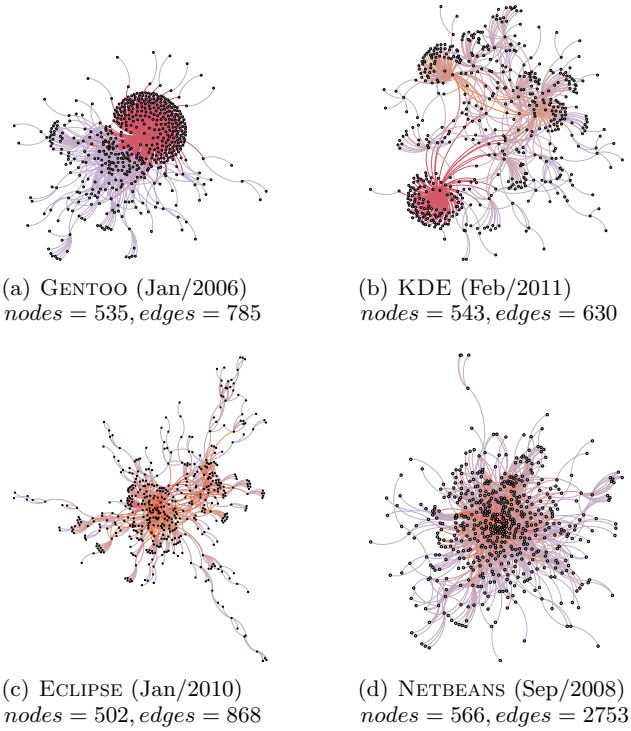


(a) Nodes in LCC (b) Assortativity (c) Clustering coeff. (d) Closeness central.

(e) Nodes in LCC (f) Assortativity (g) Clustering coeff. (h) Closeness central.

Figure 1 Evolution of structural measures of the LCC in the monthly Bugzilla collaboration networks. (a-d): Gnu/Linux related projects Gentoo (black) and KDE (green), (e-h): IDEs Eclipse (black) and NetBeans (green).

A different perspective of the structural change the Gentoo community was undergoing is given in Figure 1(d) which displays a visible plateau in the closeness centralisation of the network within the same period. In fact, as can be seen in the network depicted in Figure 2(a), in the period between 2006 and 2008 most of the collaborations were mediated by a



(a) GENTOO (Jan/2006)
$nodes = 535, edges = 785$

(b) KDE (Feb/2011)
$nodes = 543, edges = 630$

(c) ECLIPSE (Jan/2010)
$nodes = 502, edges = 868$

(d) NETBEANS (Sep/2008)
$nodes = 566, edges = 2753$

Figure 2 Four monthly collaboration networks with comparable size showing largely different social organisation (the network visualisation was generated by GEPHI [1]).

single central community member, while the social organisation of the ECLIPSE community depicted in 2(c) was structured in a much more homogeneous way. The evolution of degree assortativity is captured in Figures 1(b) and 1(f). Both the level of degree assortativity as well as its dynamics differ across the projects. The collaboration network of ECLIPSE exhibits a tendency towards assortative structures (meaning that high degree nodes are preferentially connected to high degree nodes). The opposite is true for the KDE and the GENTOO communities which show a tendency towards disassortativity. We thus argue that assortativity is suitable to further differentiate the social organisation of OSS communities.

4 Conclusions and Future Work

We have studied measures that capture different structural dimensions in the social organisation of OSS projects. Our analysis is based on a comprehensive dataset collected from the bug tracking communities of 14 major OSS projects. We view the social organisation from the perspective of time-evolving networks and highlight how projects, although similar in terms of size, problem domain and age, a) largely differ in terms of clustering coefficient, assortativity and closeness centralisation and b) that some projects show interesting dynamics with respect to these measures that cannot be explained by mere size effects. We argue that the phase of high closeness centralisation and low clustering coefficient observed in the GENTOO community between 2006 and 2008 may be interpreted as a lack of *social cohesion* which can possibly pose a risk for the project.

While our results are necessarily preliminary, we currently extend our work by adding spectral measures like algebraic connectivity and inequality measures like the Gini index

that can highlight further differences in the social organisation [13]. A detailed case study is under preparation [14] and further includes community performance indicators (e.g. response times, bug fixing times and fraction of open issues) that can be mined from our dataset. The eventual goal of our project is the provision of multi-dimensional indicators for the social and technical organisation of OSS projects that are correlated with performance and that can be considered in the management and evaluation of OSS projects [12, 15, 10]. Such indicators can be useful when taking informed decisions about which OSS project to invest in or rely on. Furthermore, due to the distributed nature of collaborations, individuals often lack a global perspective on evolving communication and coordination structures, even though these can influence long-term success. An inclusion of suitable indicators in community platforms like e.g. Bugzilla can assist in determining risks and allow project managers to timely react by shifting responsibilities, fostering information flow or changing organisational procedures.

References

1 M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks. In *Proceedings of the ICWSM '09*. AAAI, 2009.

2 Linton C. Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1979.

3 James Howison, Keisuke Inoue, and Kevin Crowston. Social dynamics of free and open source team communications. *Open Source Systems*, pages 319–330, 2006.

4 Mark E. J. Newman. Mixing patterns in networks. *Phy. Review E*, 67:026126, 2003.

5 Mark E. J. Newman. *Networks: an introduction*. Oxford Univ Press, 2010.

6 Roozbeh Nia, Christian Bird, Premkumar Devanbu, and Vladimir Filkov. Validity of network analyses in open source projects. In *proceedings of the 7th IEEE Working Conference on Mining Software Repositories (MSR)*, pages 201–209. IEEE, 2010.

7 Eric S. Raymond. The cathedral and the bazaar. *Knowledge, Technology & Policy*, 12(3):23–49, 1999.

8 Gregorio Robles and Jesus Gonzalez-Barahona. Contributor turnover in libre software projects. In Ernesto Damiani and et al, editors, *Open Source Systems*, volume 203, pages 273–286. Springer Boston, 2006.

9 Nicolas. Serrano and Ismael Ciordia. Bugzilla, itracker, and other bug trackers. *Software, IEEE*, 22(2):11–13, 2005.

10 Claudio Juan Tessone, Markus Michael Geipel, and Frank Schweitzer. Sustainable growth in complex networks. *Europhysics Letters*, 96:58005, 2011.

11 Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

12 Marcelo Serrano Zanetti. The co-evolution of socio-technical structures in sustainable software development: Lessons from the open source software communities. In *proceedings of the 34th ICSE, doctoral symposium*, pages 1587–1590, 2012.

13 Marcelo Serrano Zanetti, Ingo Scholtes, Claudio Juan Tessone, and Frank Schweitzer. The evolution of social organisation and risk in open source software projects. In preparation.

14 Marcelo Serrano Zanetti, Ingo Scholtes, Claudio Juan Tessone, and Frank Schweitzer. A quantitative study of social organisation in the gentoo community. In preparation.

15 Marcelo Serrano Zanetti and Frank Schweitzer. A network perspective on software modularity. In *GI-Edition - Lecture Notes in Informatics (LNI), Proceedings P-200, ARCS 2012 Workshops*, pages 175–186, 2012.

16 Minghui Zhou and Audris Mockus. What make long term contributors: Willingness and opportunity in oss community. In *proceedings of the 34th ICSE*, pages 518–528, 2012.

Apply the We!Design Methodology in E-learning 2.0 System Design: A Pilot Study

Lei Shi, Dana Al Qudah, and Alexandra I. Cristea

Department of Computer Science, University of Warwick
Coventry, CV4 7AL, UK
{lei.shi, dqudah, acristea}@dcs.warwick.ac.uk

Abstract

During the emergence of Web 2.0, the methodologies and technologies of E-learning have developed to a new era, E-learning 2.0, emphasises on social learning and the use of social interaction tools. The students are the main end-user of the E-learning 2.0 systems, so it is essential to take students' opinions into consideration during the design process of such systems. The We!Design participatory design methodology is proposed for incorporating undergraduate students in the development of educational systems. This pilot study aims to investigate how the We!Design methodology would work and what the results might propose, and gather initial preferences and improve the quality and efficiency of the larger scale studies in the future.

1998 ACM Subject Classification D.2 Software Engineering

Keywords and phrases Participatory design, Requirement analysis, E-learning 2.0, Web 2.0

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.123

1 Introduction

The We!Design is a student-centred educational system design methodology, which supports typical content-independent educational processes and can be easily applied in real educational contexts [1]. Undergraduate students are the main end-users of the e-learning systems, and they have substantial abilities to propose the problems and even the solution to the problems according to their e-learning experience. Besides, undergraduate students, especially those who are studying computer science, are willing to participate in the process of educational system design [2]. The design process involves both the system designers and the students, and provides a tool to exchange knowledge between them [3], hence helps system designers gather the potential end-users' real needs.

As one of the participatory design methodologies [4][5], the We!Design engages undergraduate students, the potential end-users of the results of the design activities, as important participants in the design process. With the coordination of coordinator(s), the students participate in the design tasks and make design decisions by cooperating and discussing. Comparing to other participatory design methodologies, the We!Design methodology 1) requires a short period of time of cooperation between designers and students, which makes it easier to involve and motivate students; 2) towards the design of learning systems rather than learning content, which supports content-independent educational process, such as note-taking and various forms of assessment; 3) exploits the design competencies of highly computer-literate students rather than the participation of the students with average technological knowledge, which is conductive for the students to contribute to the user interface prototype design in an efficient manner [6]. For these reasons, we choose the We!Design as the participatory design methodology in our research.

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 123–128

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

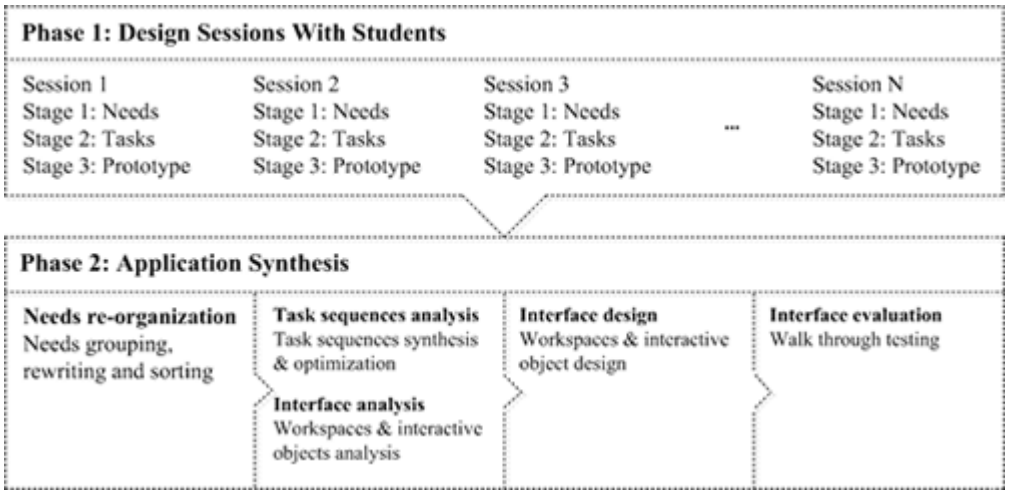


Figure 1 The We!Design Methodology [1].

This pilot study conducts the We!Design methodology in a small scale experiment. The main goals of this study presented in this paper are: 1) exploring how the We!Design methodology would work and what the results could be proposed; 2) gathering initial preferences and improving the quality and efficiency of the larger scale experiments in the future. The remainder of the paper is organized as follows. Section 2 describes the process of the experiment applied the We!Design methodology and the experiment results. Section 3 discusses the problems occurred in the experiment and the possible solutions. Section 4 draws the conclusions.

2 Experiment

As shown in Figure 1, the We!Design methodology contains 2 phases. In the first phase, several parallel design sessions take place with small groups of students. Each design session is conducted with the coordination of coordinator(s), in order to guide the students and facilitate their collaboration during the whole session. This session includes 3 stages, needs collecting, tasks sequencing and prototype. By going through the session, a requirements list and a low-tech prototype are expected to be proposed. In the second phase, the system designers analyse the results proposed in the first phase and synthesize them in a single system with an ordered requirements list [6].

In this study, the experiment was conducted with the participation of 2 coordinators and 6 fourth year undergraduate students. One coordinator was a computer science Ph.D. from the University of Nottingham; the other coordinator was a computer science Ph.D. student from the University of Warwick. The students were from the Computer Department of 'Politehnica' University of Bucharest, Romania, studying a course on 'Semantic Web'.

A short seminar was conducted at the beginning of the experiment, in order to introduce the experiment process, explain the experiment goals, and recall the required background knowledge such as how to design a system and what an e-learning system is. With some case studies of e-learning systems, the students became more confident to discuss and present their ideas, so the coordinators could focus on time controlling and summarizing.

2.1 Phase 1: Design Sessions with Students

Two parallel design sessions were conducted in the first phase. Each design session involved 2 coordinators and 3 students, and lasted for about 2.5 hours. One coordinator was a human computer interaction (HCI) expert preserving the usability of the system; the other coordinator was an e-learning system expert preventing the students from going too far away from the system design goals. Besides, both coordinators were also in charge of guiding and facilitating the students to go through the session, and providing support without interfering in the process of decision making. In front of the students, there was a table with pens and big white paper for the students to record their ideas and draw the user interface of the prototype. The experiment process was recorded by a video camera, so the coordinators could focus on guiding the experiment and noting the problems occurred [1].

In the first stage, the students were asked to extract a set of needs for the new e-learning system based on their experience of using such systems. Initially, the students proposed the important features that they expected to be provided by an e-learning system and the problems they found during using e-learning systems. Next, they summarized all the ideas into a needs list, and continually elaborated, categorized and evaluated these needs. Finally, 97 raw needs were proposed and ordered into a requirement list according to their importance.

The second stage aimed to satisfy the needs proposed in the first stage, by describing the interactions between end-users and the system. *Personas* and *scenarios* [7] were adopted to describe the process. A persona is a 'hypothetical archetype' of an actual user. S/he is not a real person, but is used to represent a real persona in the design process [8]. A *scenario* is a description of a *persona* who is using a system to fulfil several tasks in a specific context to achieve goals [9]. 4 personas were created to represent the types of users in the use scenarios. One of the examples designed by the students is:

Bob is a freshman, taking the module of 'Java programming language'. He hasn't learnt any other programming languages before, so there are several concepts that he doesn't understand, such as the reason of Object Serialization and the difference between Interfaces and Abstract Classes. He is keen to get the answers by discussing with his friends rather than reading a chapter. He finds Alex has lots of programming experience, so he decides to ask Alex. Bob sends a message to Alex to describe his problems. Alex describes his understanding about Object Serialization. And then Bob asks him to send some coding examples.

At the end of the second stage, one student in the group presented the task sequences, while the HCI expert coordinator wrote down the task sequences during the discussion.

In the third stage, the task sequences were refined and converted into more concrete system requirements. The students were asked to identify the key features of the task sequences, in order to sketch out the layout and the user interface for the low-tech prototype on the big white paper (shown in Figure 2). At last, a stereotypical role-play testing was conducted, to evaluate the usability and note the problems and potential solutions.

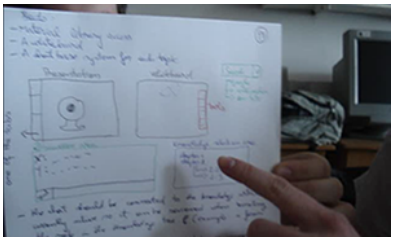


Figure 2 User interface of the prototype for the e-learning system.

2.2 Phase 2: Application Synthesis

In the second phase, the requirements proposed during the design sessions were synthesized into the final system [6]. Firstly, the system designer gathered the requirements from the two sessions, grouped the similar requirements, and removed the duplicates. Subsequently the importance of these requirements was estimated according to the number of times the requirement appeared in the design sessions and importance suggested by the students. Finally, the designer synthesised a list containing 28 requirements, ordered by the importance, and divided them into 4 categories, which represent the main areas for which features could be built within a system, as shown in Table 1.

Table 1 The Final Requirement List.

Category	Requirement	N[1)]	I[2)]
Learning	Use multiple types of files, e.g. PDFs, photos, videos, slides, etc.	5	1
	Tag and flag up topics in the learning path	1	2
	Take tests after learning a topic	4	3
	Get assessment and feedback from teachers	5	4
	Access to open learning resource, e.g. Wikipedia	6	5
	Search learning resource within and outside of the system	6	6
	View learning progress in percentage	5	7
	Contribute to learning materials by creating and uploading files	3	8
	Choose to view the whole or partial learning path	1	9
Social Networking	Create groups that are registered for the same topic	3	1
	Share and/or recommend learning materials	2	2
	Ask and answer questions of other students	5	3
	Discuss the current learning topic with other students	6	4
	Use feedback & questions forum at the end of each lesson	5	5
	Use communication tools to chat and leave messages	4	6
	Write comments/notions wherever and whenever they want	5	7
	Create groups that share common learning interests	4	8
	View history discussion when selecting a particular topic	1	9
Adaptation	Recommend topics according to student's knowledge level	4	1
	Recommend other topics according to the current learning topic	5	2
	Recommend topics by referring to other students' rating	2	3
	Adapt learning path according to learning progress	2	4
	Adapt learning tools according to student's user-level	1	5
Usability	Use graphical user interfaces	4	1
	Get instructions and tips	3	2
	View system status	2	3
	Select full screen option	1	4
	Set themes, layout, etc.	2	5

1) N: The number of times the requirement appeared in the students' suggestions, in one form or another;
2) I: The average importance of the requirement proposed by the students from the two design sessions.

3 Discussions

The *cold-start* problem appeared as expected. At the beginning of the experiment, the coordinators explained the process and goals of the experiment, and introduced some required background knowledge followed by several case studies, but it was still not easy to

get the students started, because they were afraid of proposing something that might not make sense. Therefore, the coordinators should have the ability to recognize the students' problems and find out good solutions and encourage them to participate in the discussion and presentation. One feasible method is to ask some open-ended questions and give some typical answers, so the students can realize what kinds of questions and answers are appropriate.

In the needs collecting stage, the students tended to explore the solutions to satisfy the needs as well, but the objective of this stage is to focus on needs collection rather than to find the solutions. Hence the coordinators should remind them in appropriate way and stop them in time. *Personas* and *scenarios* were adopted in the stage of tasks sequencing. It is necessary to keep in mind that people are diverse; they have different experiences, different expectations and different preferences, so it is difficult to design for all of them. The solution Cooper proposed is to identify the *primary persona* as the individual "whose needs must be met, but whose needs cannot be met through an interface designed for any other personas" [9][10]. *Scenarios* are short, fictional stories that describe a set of tasks and interactions of the *personas*. The more-detailed scenarios can provide more information for tasks sequencing, but due to the short period of time, the coordinators should guild the students to design an appropriate level of detail. In the prototyping stage, some design flaws were founded, and the students might be reluctant to fix them or need extra time. The coordinators should encourage them to fix the flaws as well as control the time, because even incomplete work can still help to inspire the system designers.

In the application synthesis phase, an ordered requirement list was proposed in a generic detail level, which means it is necessary to generate the requirements specification (intermediate detail level) and then the application specification (high detail level) in the next step [11]. Besides, the system requirements were arranged by the system designers, according to students' content-based descriptions, so it is possible for the designers to misunderstand students' intention. Therefore, it is necessary to ask the students to check the consistency between the reorganized requirements and their original ideas.

4 Conclusions

In this paper, we have applied the We!Design participatory design methodology in a small scale experiment for a pilot study. Two coordinators and six computer science undergraduate students were involved in the experiment. Two parallel design sessions were conducted in the first phase. The students went through the stages of needs collecting, task sequencing and prototype designing, and proposed a requirement list and a low-tech prototype. In the second phase, the system designer synthesised the requirements proposed in the first phase into the system requirement list, categorized them according to the features that could be built within a system, and sorted them according to the importance.

We discussed the problems occurred during the experiment process and investigated the possible solutions. The key to better conduct the experiment is to encourage the students to participate in discussion and presentation. Due to the lack of time, the coordinators should keep the balance between the detail level of discussion and time controlling, and it is better that they provide some tools and tips during the experiment, e.g., *personas* and *scenarios*. We also discussed the importance of mutual understanding between the system designers and the students. A feasible way is to ask the students to check the consistency.

This pilot study helped us to explore the requirement analysis experiment applied the We!Design participatory design methodology. A much larger study will be conducted in the future to analyse the system requirement for a real e-learning system design.

References

1 Triantafyllakos, G., Palaigeorgiou, G., Demetriadis, S. and Tsoukalas, I.A. The We! Design Methodology: Designing Educational Applications with Students, In *Proceedings of the 6th International Conference on Advanced Learning Technologies*, pages 997-1001, 2006.

2 Siozos P., Palaigeorgiou G., Triantafyllakos G. and Despotakis T. Computer based testing using 'digital ink': participatory design of a Tablet PC based assessment application for secondary education. *Computers & Education*, 52(4):811–819, 2008.

3 Roda, C. Participatory system design as a tool for learning, In *Proceedings of IADIS International Conference of Cognition and Exploratory Learning in Digital Age*, pages 366-372, 2004.

4 Muller, MJ. and Druin, A. Participatory design: the third space in HCI. In *Handbook of HCI. Edited by Muller MJ.*, pages 1-31, 2003.

5 Kyng, M. Bridging the gap between politics and techniques: On the next practices of Participatory Design. *Scandinavian Journal of Information Systems*, 22(1): 49-68, 2010.

6 Triantafyllakos, G. N., Palaigeorgiou, G. E., and Tsoukalas, I. A. We!Design: A student-centered participatory methodology for the design of educational applications. *British Journal of Educational Technology*, 39(1):125-139, 2008.

7 Cooper, A. and Reinman, R. *About Face 2.0: The Essentials of Interaction Design*. John Wiley & Sons, Inc., 2003.

8 Calabria T. An introduction to personas and how to create them. Captured in: http://www.steptwo.com.au/papers/kmc_personas/index.html, viewed 8/07/2012.

9 Cooper, A. *The Inmates Are Running the Asylum*. Indianapolis, Indiana: SAMS, A Division of MacMillan Computer Publishing, 1999.

10 Cooper, A., Reimann, R., and Cronin, D. *About Face 3, The Essentials of Interaction Design*. Published by Wiley Publishing, Inc., 2007.

11 Sommerville, I. and Sawyer, P. *Requirements Engineering: A Good Practice Guide*. John Wiley & Sons, Inc., 1997.

An Implementation Model of a Declarative Framework for Automated Negotiation

Laura Surcel

University of Craiova
Blvd. Decebal, nr. 107, RO-200440, Craiova, Romania
laura_surcel@yahoo.com

Abstract

The subject of automated negotiations has received a lot of attention in the Multi-Agent Systems (MAS) research community. Most work in this field on the auction design space, on its parametrization and on mechanisms for specific types of auctions. One of the problems that have been recently addressed consists in developing a generic negotiation protocol (GNP) capable of governing the interaction between agents that participate in any type of auction. Though much has been said on this matter, the current results stop at the XML representation of specific negotiation mechanisms. In this paper we propose a declarative approach for specifying a generic auction protocol by using Belief-Desire-Intention (BDI) agents and the Jason programming language to represent the entities that communicate in an auction. In order to validate the claim on the generality of the proposed approach we have used the GNP to model two negotiation mechanisms: one for the English auction and one for the Dutch auction.

1998 ACM Subject Classification I.2.11 Distributed Artificial Intelligence

Keywords and phrases Auctions, automated negotiations, multi-agent systems, Jason, generic negotiation protocol

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.129

1 Introduction

Generally speaking, a negotiation is a bargaining (give or take) process between two or more parties (each with its own aims, needs and viewpoints) seeking to discover a common ground and reach an agreement to settle a matter of mutual concern or resolve a conflict[1]. Negotiation is employed in many areas, such as: law, business, air and marine traffic management, trade, parenting, hiring a.o.

A notable subfield of negotiations is represented by auctions which are also the focus of the present paper. An auction is a process of exchanging possessions through buying and selling by offering them up for bid. A person or an organization may use auctions to sell goods or services by making them available to the public, setting an initial price and then receiving offers. In the general case the winner of the auction is the buyer that makes the greatest bid.

The importance of this subject is supported by the popularity and the use of hundreds of web sites that have transformed auctions into an open process, in which thousands of items may be offered for bidding by anyone from anywhere at any time[2]. All these sites offer the

[1] http://www.businessdictionary.com

[2] The five most reliable, renowned and diversified online auction sites with respect to user options, according to a 2012 survey conducted by TopTenReviews (http://online-auction-sites.toptenreviews.com/), are: eBay, WebStore, eBid, OnlineAuction and OZtion

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 129–134

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

possibilities of selling and buying which must be performed by human operators. This last observation leads to some interesting questions that form the starting point of the current research: is there a way to automate the negotiation process? Can negotiations be carried out between computers with minimum human input? And can such a process be universally applied to any kind of auction?

Researchers [1, 2] that tried to answer these questions reached the following results: any negotiation can be considered as an interaction between a mechanism and a strategy. The mechanism (or protocol) is a set of rules that must be followed by participants in order to communicate and it is public, while the strategy describes the behavior of a negotiation participant and it is used for reaching his or her private goals [1]. Although many models of automated negotiations such as: bargaining, auctions[4, 5], multi-commodity negotiations a.o. have been brought forward, this paper discusses the model proposed in [1] and describes a possible extension of the GNP from an English auction to a Dutch auction using the Jason agent programming language [6].

Our choice is justified by the fact that [1] identifies a minimal generic protocol and a declarative approach of representing rules and constraints specific to negotiation types, it highlights a set of basic concepts that could be part of a core negotiation ontology and it presents the initial prototype for the English auction which served as a starting point for the present implementation. With respect to the chosen programming language, there were three reasons that made its employment appropriate: (1) Jason supports the declarative programming paradigm, closer to logic programming; (2) it is implemented in Java which makes it multi-platform; (3) a Jason multi-agent system can be easily distributed over a network (for example, by using JADE).

The paper is structured as follows. In Section 2 we outline the context that led to this approach of a GNP. Section 3 demonstrates the specific approach by providing code samples and by offering details on how the initial prototype from [1] was extended. Finally, Section 4 summarises the results and presents the conclusions.

2 Background and Related Work

Most of the papers that deal with the subject of automated negotiations are focused on describing specific protocols for different kinds of interactions. However, in what follows, we will only produce a brief description of those that tackle the problem of a generic protocol and that have influenced the present work.

Article [3] presents an electronic market architecture whose main asset is a declarative auction language (DAL) expressed by means of an XML representation. Rolli *et al.* claim that such a language makes possible the design of auction mechanisms, code generation, deployment of market instances and modelling of participants' behaviour. Albeit its expressiveness, the computational complexity of the model is a real drawback especially when there are introduced more mechanisms. On the other hand, using an interpreter for declarative programming languages like Jason is more efficient. Additionally, its similarity to logic programming simplifies the process of specifying new negotiation mechanisms.

Article [2] opens the way for a universal protocol by proposing a generic software framework for automated negotiation. Bartolini *et al.* argue that the presented interaction protocol is simple enough and it can be used in all circumstances. Moreover, a taxonomy for negotiation rules is presented, which identifies and outlines the different roles of agents within an auction. Although very helpful for this research, the illustrated approach is relatively limited because it only addresses the problem from the perspective of the negotiation host, i.e. the agent

that controls the negotiation process. Nevertheless, the paper remains important because the proposed taxonomy has been incorporated into the declarative framework in order to delimitate between the different stages of an auction.

Article [1] represents the foundation for the present implementation model since it addresses the problem from the perspective of both the auction host and the participant. The authors bring to the forefront of the architectural model three types of agents:

- *Auction Service (AS)* It manages the auction related activities like: auction creation, auction termination and it keeps track of all the current auctions by containing an auction directory;
- *Auction Host (AH)* (see Sec. 3.1);
- *Auction Participant (AP)* (see Sec. 3.2).

Another interesting observation of the authors is that, from a conceptual point of view, the above agents can be described by means of the following equations:

$$AH = GNP_{rolehost} + DNM_{host}$$
$$AP = GNP_{roleparticipant} + DNM_{participant} + CNS$$

where:

- **GNP** is the Generic Negotiation Protocol that governs the interaction between agents;
- **DNM** is the Declarative Negotiation Mechanism (see Sec. 3.3);
- **CNS** represents the Custom Negotiation Strategy and it is specific to a given AP as it can be noticed from the second equation. Nevertheless, the CNS must be consistent with the DNM. Its purpose is to help the agent take the appropriate steps (that remain within the limits of the DNM) in order to achieve its own aims[3].

Since the proposed prototype was rather simple and illustrated only the workflow of an English auction, the general character was not fully achieved. As a result, we have set the goal for this paper the extension of its applicability. In what follows, we will discuss the changes on the original model and the improvements that we propose.

3 Implementation Details

From the three agent types specified in the previous section, only the AH and the AP will be presented at large, since they illustrate better the improvements brought to the model proposed in [1]. Nevertheless, one important observation must be made on the AS: it creates a new auction instance (AIN) based on a short description (which consists of the auction type and the product name) offered by the auction initiator participant (AIP). This can be considered the first situation of auction parametrization because the AIN is informed of the specific type of auction that it will manage.

3.1 Auction Host

The auction host is a unique agent per auction instance and it represents the authority that governs the auction. According to [2], AH plays the following roles: gatekeeper (decides which agents can be submitted to negotiation), proposal validator (verifies if a proposal

[3] For more details on the architecture please consult [1]

satisfies the negotiation template), protocol enforcer (determines the circumstances in which a participant may post a proposal), information updater (changes the parameters of the negotiation as it unfolds), negotiation terminator (specifies when no more proposals may be posted) and agreement maker (chooses from a set of valid proposals the one (those) that should be turned into an agreement(s)).

Due to the fact that Jason agents pursue their goals by applying plans which are compliant with their beliefs and rules (for more details see reference [6]), the definition of an AH's behavior comes easily.

```
+register[source(A)]
    : can_register(A)
    <- +registered(A);
       ?buy_it_out(Sum);
       ?increment(Increment);
       ?items(Left_items);
       ?last_offer(Offer);
       ?state(State);
       .send(A, tell, registered(info(Sum,Increment),
        status(Offer, Left_items, State))).
```

```
+fold[source(A)]
    : registered(A)
    <-  -registered(A);
        .send(A, tell, not_registered).
```

Listing 1 The Auction Host – gatekeeper role implementation.

In order to illustrate its role as a gatekeeper the *register* and *fold* plans were considered (see listing 1). That is, every time the AH receives a request of registration or withdrawal from an AP, it applies one of the implemented plans based on some conditions (can_register(A), registered(A)). The conditions may vary from one auction to the other so they are specified by the DNM. The other roles of the AH have been outlined through a set of plans and goals illustrated in listing 2.

```
+bid(Offer,Items)[source(A)]
    : check_protocol(A)
    & check_proposal(Offer,Items)
    & not(terminate(Offer,Items))
    <- -+state(processing);
       !update_status(A,Offer,Items);
       !inform_participants(A,Offer);
       -+state(bidding).
```

```
+!close
    : check_winner
    <- ?initiator(I);
       -+state(closed);
       ?bidders(A);
       .send(I, tell, winner(A));
       .send(A, tell, winner).
```

Listing 2 The Auction Host – proposal validator, protocol enforcer, information updater and negotiation terminator roles' implementation.

The modifications made with respect to the initial model are the following: the "buy-out" and the fold options were introduced and new goals were created (update_status and inform_participants) in order to outline the different roles of the AH and also the phases of an auction.

3.2 Auction Participant

The AP only registers for an auction, receives a minimal description of it (which was not present in [1]) and, based on that description, it chooses the strategy that bests suits its goals. In this case, there have been implemented three kinds of strategies: firstly, if there is a "buy-out" sum and it is public (the Sum parameter must be a non-zero number), then it offers the sum immediately (aggressive bidder); secondly, if there is no "buy-out" sum and it is an ascending auction (Increment is positive), then strategy one is applied, i.e. it continuously bids and increases the last offer by Increment units until the auction closes or it is declared winner; thirdly, if there is no "buy-out" sum and it is a descending auction

(Increment is negative), then strategy two is applied, i.e. as long as the last accepted bid is greater than its available amount of money it remains idle and when the offer is lowered enough it starts bidding by decreasing its last offer until the auction closes or it is declared winner. Listing 3 displays the selection of the second strategy (which follows the rules of an English auction), the others being similar. We also present the implementation of the third strategy that conforms to the rules of a Dutch auction.

Note that selecting a suitable bidding technique only depends on whether there is a "buy out" sum and whether the increment is positive or negative which naturally leads to a poor strategy. Additionally, the type of AP displayed here is that of an aggressive bidder.

```
+registered(info(Sum,Increment),
status(Offer, Left_items, _))
    : Sum==0 & Increment>0
    <- -+current_quote(Offer);
       +increment(Increment);
       -+items(Left_items);
       +strategy(1);
       !bid_strategy1.
```

```
+!bid_strategy2
: not(accepted) & amount(Amount) &
  items(No) & No>0 & current_quote(Quote) & Quote<=Amount
<-  ?auction_host(AuctionHost);
   -+current_offer(Quote);
   .send(AuctionHost, tell, bid(Quote, 1)).
```

Listing 3 The Auction Participant - strategy determination and implementation.

3.3 Declarative Negotiation Mechanisms for English and Dutch Auction

The DNM depends on the type of negotiation and it is used for customizing the GNP. It is a layer of the architecture which differentiates between auction types. In the present implementation model it consists of a set of rules (and goals) that the AH applies (and pursues) during the course of an auction and it follows the taxonomy proposed in [2]. This classification identifies conditions for: admission of participants (can_register(A)), proposal validity (check_proposal(Offer,Items)), protocol inforcement (check_protocol(A)), updating status and informing participants (update_status(A,Offer,Items), inform_participants(A,Offer)), lifecycle of negotiation (terminate(Offer, Items)), agreement formation (this part of the auction has not been implemented yet but it is considered for future work). The rules and goals that differ the most between the two auction types are illustrated by listing 4:

```
// Dutch auction
check_proposal(Offer,Items)
    [state(bidding)] :-
    asked_price(AskedPrice) &
    items(Left_items) &
    Left_items>=Items &
    Offer <= AskedPrice.

+!update_status(A,Offer,Items)
    <- .time(H,M,Sec2);
       ?items(I);
       ?increment(Increment);
       ?asked_price(AskedPrice);
       ?bidders(B);
        -+last_offer(Offer);
        -+items(I-Items);
        -+bidders([A|B]);
        -+asked_price(AskedPrice+Increment);
        -+last_update(Sec2).
```

```
// English auction
check_proposal(Offer,Items)
[state(bidding)] :-
    increment(Increment) &
    last_offer(Quote) &
    items(Left_items) &
    Left_items>=Items &
    Offer >= Quote + Increment.

+!update_status(A,Offer,Items)
    <- -+last_offer(Offer);
       -+bidders([A]).
```

Listing 4 Dutch versus English auctions DNM.

4 Conclusions

In this paper we have presented a new version of the prototype of a GNP proposed in [1]. The improvements relate mainly to AH and AP implementation. On the one hand, the AH displays a better separation of roles which also leads to a better organization of an auction's workflow. On the other hand, the AP may choose from different strategies the one that complies with the current rules based on a description of the auction and not its name. Thus, there is no need to create one agent for each kind of auction.

We believe that this approach to a declarative specification of a GNP may bring satisfactory results in the pursuit of creating a generic framework for automated negotiations. However, the current results suggest that the more comprehensive the protocol becomes, the harder it is to determine a variety of negotiation strategies. A compromise may consist in simple strategies that fail to use all the choices that a type of auction offers. We think that this is an interesting direction that should be pursued in future research.

References

1 Alex Muscar, Costin Badica. *Exploring the design space of a declarative framework for automated negotiation: initial considerations (**in press**)*. AIAI'2012 Proceedings, 2012

2 Claudio Bartolini, Chris Preist, Nicholas R. Jennings. *A generic framework for automated negotiation*. In: R. Choren, A.F. Garcia, C.J.P. de Lucena, A.B. Romanovsky (eds.) SELMAS, *Lecture notes in computer science*, vol. 3390, pp. 213-235, Springer, 2004

3 Daniel Rolli, Stefan Luckner, Henner Gimpel, Christof Weinhardt. *A descriptive auction language*. In: *Electronic Markets*, vol. 16, issue 1, pp. 51-62, Routledge, 2006

4 Peter R. Wurman, Michael P. Wellman, William E.Walsh. *A parametrization of the auction design space*. Games and Economic Behavior 35(1-2), 304-338, 2001

5 Adriana Dobriceanu, Laurentiu Biscu, Amelia Badica, Costin Badica. *The design and implementation of an agent-based auction service*. IJAOSE 3(2/3), 116-134, 2009

6 Rafael H. Bordini, Jomi F. Hubner, Michael Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*. Wiley, 2007

Blurring the Computation-Communication Divide: Extraneous Memory Accesses and their Effects on MPI Intranode Communications

Wilson M. Tan and Stephen A. Jarvis

Performance Computing and Visualisation Group
Department of Computer Science
University of Warwick, United Kingdom
Email: wilson.tan@warwick.ac.uk

Abstract

Modern MPI simulator frameworks assume the existence of a Computation-Communication Divide: thus, they model and simulate the computation and communication sections of an MPI Program separately. The assumption is actually sound for MPI processes that are situated in different nodes and communicate through a network medium such as Ethernet or Infiniband. For processes that are within a node however, the validity of the assumption is limited since the processes communicate using shared memory, which also figures in computation by storing the application and its associated data structures.

In this work, the limits of the said assumption's validity were tested, and it is shown that Extraneous Memory Accesses (EMAs) by a compute section could significantly slow down the communication operations following it. Two general observations were made in the course of this work: first, more EMAs cause greater slowdown; and second, EMAs coming from the compute section of the processes containing the MPI_Recv are more detrimental to communication performance than those coming from processes containing MPI_Send.

1998 ACM Subject Classification Modeling techniques

Keywords and phrases High performance computing, Message passing, Multicore processing, Computer simulation, Computer networks, Parallel programming, Parallel processing

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.135

1 Introduction

Many current MPI simulator frameworks such as BSIM[10], WARPP[5] and SST-Macro[7] work on the assumption of a *Computation-Communication Divide*. This assumption states that a message passing program could be divided into two components, the computation section and the communication section, each of which could be simulated independently of each other. This assumption is currently being applied both to processes that are within a single node, and those that are located in different nodes.

The Computation-Communication Divide is actually a reasonable assumption for processes that are between nodes: computation would be the program segments that a processor would handle, while communication would be taken care of by the NIC, routers, and interconnects. The two segments perform independently of each other.

For processes that are in the same node however, the divide is not as clear-cut. The issue lies with the fact that in contrast to internode communication that relies on a dedicated interconnect such as Ethernet or Infiniband, intranode communication relies on shared memory. Aside from facilitating intranode communication, the memory subsystem is also

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 135–141

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

involved in the process of computation: it stores data and instructions for the application being executed.

Given the importance of intranode MPI in future systems, it is of interest to many to know up to what extent currently held assumption about intranode communication holds. This is the focus of this work: in particular, it investigates how Extraneous Memory Accesses(*EMAs*) affect the communication between MPI processes executing in a single node setup. By "Extraneous Memory Accesses", the authors mean memory accesses that do not figure in the communication process itself, and thus involve data structures that are not being sent between processes. Two possible locations from which EMAs could come from are tested: the sending process and the receiving process.

The rest of the paper is organized as follows: the methodology and system used are described in Section 2, the results are presented and discussed in Section 3, and paper is concluded in Section 5.

2 Methodology

A custom microbenchmark called *PaMPIck* was developed for this work. PaMPIck is primarily based on a simple loop enclosing a send-receive operation pair between Process 0 and Process 1. The loop iterates 100 times. The data being transferred in each send-receive is composed of the first 100 elements of a 1 million element integer array. Each process has 2 1-million member integer arrays: a send array and a receive array. Given that each integer is 4 bytes, 400 bytes were transferred in every iteration, from the sending process' send array to the receiving process' receive array. Other sizes for the data being transferred(aside from 100 elements) were tested, but were not included in this paper for conciseness. The data being transferred between the sending process and the receiving process is never changed between send-receive iterations. The values for the 100 elements are assigned before the very first send-receive, but never changed after that. In a way, this is to actually encourage maximum cache reuse in the send-receive pair.

```
for(iterator = 0; iterator < 100; iterator = iterator + 1)
{
    MPI_Barrier
    PAPI_start(EventSet)

    if (my_rank==0) MPI_Send
    else MPI_Recv

    PAPI_read(EventSet, values)

    //---accumulation of PAPI event
    counts from the iteration---//

    PAPI_stop(EventSet, values)
%%%%%%%---variable section, depending on setup---%%%%%%%
%%%%%%%---either---%%%%%%%
    if(my_rank == 0)
    {
        for(iterator2 = 0; iterator2 < limit;
        iterator2 = iterator2 + 1)
        {
            receivearray[iterator2] = iterator;
        }
    }
%%%%%%%---or---%%%%%%%
    if(my_rank == 1)
    {
        for(iterator2 = 0; iterator2 < limit;
        iterator2 = iterator2 + 1)
        {
            sendarray[iterator2] = iterator;
        }
    }
}
```

Depending on the setup being tested, EMAs were done between send-receive operations. Two kinds of EMAs were tested: those at the sending side(Process 0), and those at the receiving side(Process 1).

Inducing EMAs consists of changing the values of some members of the array not being used by the process for the send-receive operation: this is the receive array for Process 0(the sending side), and the send array for Process 1(the receiving side). These represent accesses done by processes participating in the send-receive on memory elements that do not figure directly with the data being sent or received: in real situations, these could correspond to intermediate or scratch variables.

Each send receive operation was measured using PAPI[9], and the following events and parameters were recorded: virtual cycle time, number of instructions, number of cycles, L1 data cache misses, L2 data cache misses and LLC(last level cache) misses. To make sure that the values being read by PAPI are accurate, each send-receive is preceded by a barrier: this is necessary so that the times and cycles being spent for doing EMAs would not reflect on the values measured by PAPI. The number of cache misses incurred during program execution is very sensitive to many factors such as other programs concurrently running in the system. Therefore, utmost care was taken to ensure that all experiments were carried out in identical conditions as much as possible.

The processor used in this study is an Intel Core i5-2430M, running at 2.40GHz. The i5 is a dual core processor, with three levels of cache memory. Each core has two 32 KB first level caches, one for instruction and one for data. The L2 cache is shared between data and instructions, and is sized at 256 KB. It is core specific. The 3MB 3rd level cache is shared among all cores in the processor.

The operating system of the platform is Linux kernel version 3.0.0-15. Programs were compiled using gcc 4.6.1, with the *-O0* optimization flag. The MPI implementation utilized was OpenMPI[4] 1.4.3, and programs were ran with the "–bind-to-core" flag.

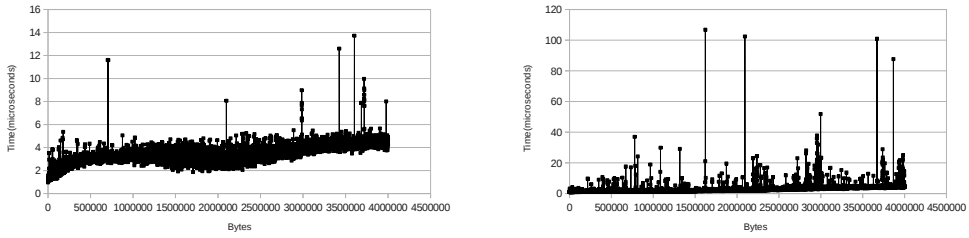
3 Results and Discussion

3.1 Latency: Send-side EMAs and Receive-side EMAs

For the EMAs coming from the Send-side and the Receive-side, several values were tested. The values ranged from no extraneous array entries(0 bytes) modified between iterations to the entire extraneous array being modified(4 Mbytes) between iterations. Values were separated by increments of 100 elements or 400 bytes, resulting in a total of 10,000 runs for each side.

The resulting average virtual/system times(or *latencies*) taken by the MPI_Recv for setup with the Send-side EMAs are shown in Figure 1a, while those with the Receive-side EMAs are shown in Figure 1b. Only the MPI_Recv data is shown primarily for the purpose of brevity. It is nevertheless definitive of the send-receive pair, since a send-receive pair could only be considered finished upon the successful completion of the receive half. Also, according to measurements, an MPI_Recv operation significantly takes up more time than its MPI_Send counterpart.

An immediate observation that could be made about the two graphs already presented is the scattering of the values: the relationship between the number of bytes modified and the operation time is definitely not linear for either setup. Nevertheless, despite the lack of a perfectly linear relationship, it is clear that the time values do tend to increase as the number of bytes modified between iteration increases. Of particular interest to us is the development of the *latency lower bound line*, or the line defining lower boundary of the region



(a) **(b)**

Figure 1 Average latencies for MPI_Recv, send-side EMA setup and receive-side EMA setup.

formed by the aggregation of data points. For instance in Figure 1b, while the average time values fluctuate, the graph shows that beyond 238,400 bytes modified between iterations, the latency would no longer go lower than a microsecond; beyond 2,337,000 bytes, it would no longer go below 2 microseconds.

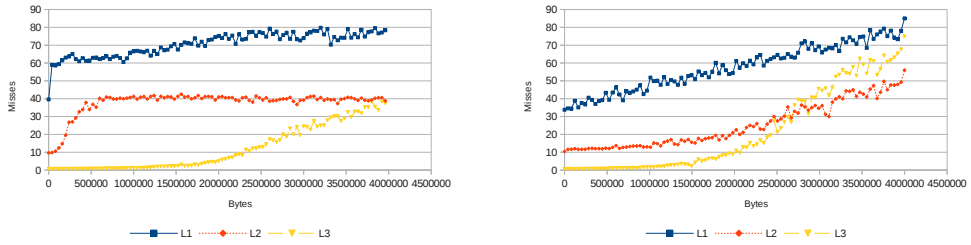
The **First General Observation** of the paper could now be stated: *while the latency of a receive operation could fluctuate up and down, there is always a lower bound value below which it will never go lower than, and that value increases as the number of bytes modified by the preceding compute section increases.*

It is interesting to note that most MPI microbenchmarks also utilize repeated send-receive pairs when measuring latency or bandwidth, not unlike what was utilized in this study. The send-receive pairs in many of these benchmarks are usually separated by very little if any computation, and thus correspond nicely with the leftmost part(bytes = 0) of Figure 1a and Figure 1b.

3.2 Cache Misses: Send-side EMAs and Receive-side EMAs

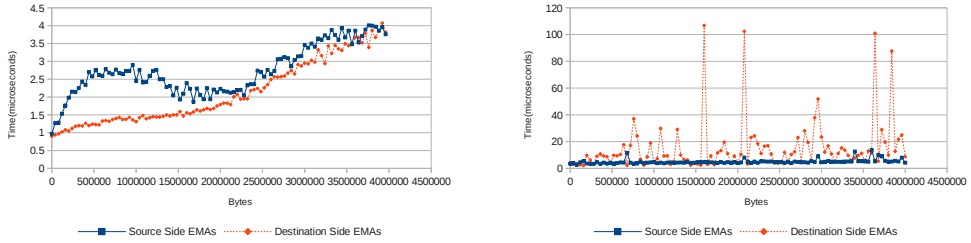
As for the underlying reason behind the general increase in latency as the the number of modified bytes increases, results indicate that the latency trend follows the trend of the average LLC or Level 3 cache miss very closely. This makes is to be expected, since the latency for cache misses in the last-level cache is several times larger than those in higher level caches[6]. Like the latency, the number of cache misses also tend to fluctuate and form dense scatter graphs. For ease of presentation, scatter graphs for the average cache misses were no longer plotted. Instead, the lower bound of the region formed by the conglomeration of data points was derived and plotted. This was done for all three cache levels. The extraction process consisted of taking the minimum of 100-data point exclusive windows, with the first window covering data points 1 to 100, the second window covering 101 to 200, etc. The lower bound lines for the 3 cache levels of the receive side are plotted in Figure 2a.

It could be observed that on the side of the receive side, the number of L1 cache misses always dominate and plateau early, followed by the 2nd level cache(Figure 2a). The number of 3rd level cache misses start rising very slowly and plateaus much later than the other two caches. Take note that the 3rd level cache is shared between all cores, so ascribing the number the misses at that level to a specific core or process with absolute certainty is difficult.



(a) (b)

Figure 2 Average cache misses for MPI_Recv, receive-side EMA setup and send-side EMA setup.



(a) (b)

Figure 3 Lower bound and Upper bound values for the average latencies of MPI_Recv.

3.3 Comparison: Send-side EMAs vs Receive-side EMAs

To be able to compare the effects of Send-side EMAs and Receive-side EMAs, the upper bound and lower bound lines of Figure 1a and Figure 1b were plotted in Figure 3a and Figure 3b. The technique was the same as the one used in Section 3.2, with the exception that the window maximum was taken for the upperbound instead of minimum. The minima(one from each side) are then plotted in Figure 3a, and the maxima in Figure 3b.

Figure 3a, shows that for an equivalent number of EMAs, those from the Receive side actually result in better(lower) bound values than those from the Send side. However, from Figure 3b, it is apparent that in many instances, the Receive side has higher upper bound values than the Send side.

Distribution-wise, the latency values produced by the Send-side EMAs(Figure 1a) are significantly more clustered than the values from the Receive-side EMAs(Figure 1b): the variance of the average latencies from the Receive-side EMAs is 2.95, for that from the Send-side is just 0.6. Nevertheless, the average of the average latencies is higher in the setup with Send-side EMAs: 3.43 microseconds, against 2.92 microseconds.

All these signify that while the lower bounds are better for setups with Receive-side EMAs, in practice, most of the latencies experienced by the send-receive pairs are far higher than the lower bound. In comparison, the lower bounds for setups with Send-side EMAs are worse, but most of the latencies experienced by the send-receive pairs are closer to lower bounds.

These results lead to this paper's **Second General Observation**: *in general, Receive-side EMAs are more detrimental to send-receive performance than Send-side EMAs; at the very least, they make the latency much less predictable than Send-side EMAs.*

4 Related Works and Future Plans

Several proposals have been put forward before with the aim of improving intranode communications. Some, such as [2] focused on user-level mechanisms, while some such as [8] focused on techniques that leverage kernel-level privileges. [3] proposed a hybrid of the two, using different mechanisms depending on message size. This paper is different from all of these works in a sense that it does not propose any modification to existing intranode communications mechanisms; instead, it studied the behavior of one specific intranode communication subsystem(that of OpenMPI), and how it compares with an assumption about it frequently made by simulator framework systems.

The closest previous work to this paper is probably [1], where separate intranode communication mechanisms were compared in terms of latency, bandwidth, and effect on cache. Part of the said work studied the effect of the data transfer mechanisms on the L2 cache. In a sense, this work is the *opposite* of that work, since while the said work focused on the effect of transfer mechanisms on the rest of the application, this paper focused on how the non-data transfer portions(or compute sections) of the application affect the communication section performance. This aspect is emphasized by the fact that in PaMPIck, sends and receives were just repeated in every iteration of the test program: the data being transferred was never changed.

As for future work, the authors intend on carrying out the study on processors that feature more than 2 cores. There is one potential source of EMA that was not tested in this work: *processes that do not figure in the send-receive pair, but are concurrently active with the pair carrying out the send-receive*. The third source of EMA was not tested in the study covered by this paper because the Core i5-2430M is a *dual*-core processor, and testing the third EMA source would have needed 3 processes. The Core i5-2430M could actually support up to four MPI processes, but we would end up oversubscribing one of the cores, so the results many not be conclusive.

In the long run, the authors hope that this work would lead to the development of a better intranode communication model that takes into account the effects of EMAs, wherever they may come from. Such a model is key to projecting optimal setups for future MPI-based HPC systems, which would probably feature more intranode communications than internode ones.

5 Conclusion

The assumption of a *Computation-Communication Divide* is very convenient when reasoning about message passing programs. Unfortunately, it is not always reasonable for cases wherein the processes are located in the same node and communicate through shared memory. Experiment results showed that memory-related activities of the compute section could significantly slow down intranode communications. Two general observations were made in the course of this work: first, the more EMAs made by the preceding compute section, the greater the slowdown for the intranode communication would be; and second, EMAs made at the side of the receiving node cause greater slowdown than those from the side of the sending node. Simply put, when the interconnect has a "memory"(because it *is* memory), the boundary *blurs*. At the level of the node, if the communication section performance is to be successfully predicted, the behavior of the preceding compute section must be taken into account.

While the failure of the Computation-Communication Divide assumption could not be tagged as the source of *all* simulation inaccuracies, the authors nevertheless recommend modellers to consider it as a "suspect" when reality-simulation discrepancies arise.

References

1. D. Buntinas, G. Mercier, and W. Gropp. Data transfers between processes in an smp system: Performance study and application to mpi. In *Parallel Processing, 2006. ICPP 2006. International Conference on*, pages 487 –496, aug. 2006.
2. Lei Chai, A. Hartono, and D.K. Panda. Designing high performance and scalable mpi intra-node communication support for clusters. *Cluster Computing, IEEE International Conference on*, 0:1–10, 2006.
3. Lei Chai, Ping Lai, Hyun-Wook Jin, and D.K. Panda. Designing an efficient kernel-level and user-level hybrid approach for mpi intra-node communication on multi-core systems. In *Parallel Processing, 2008. ICPP '08. 37th International Conference on*, pages 222 –229, sept. 2008.
4. E. Gabriel, G. Fagg, G. Bosilca, T. Angskun, J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, Ralph H. Castain, D. J. Daniel, R. L. Graham, and Timothy S. Woodall. Open MPI: Goals, concept, and design of a next generation mpi implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, page 97–104, Budapest, Hungary, 09/2004 2004.
5. S. D. Hammond, G. R. Mudalige, J. A. Smith, S. A. Jarvis, J. A. Herdman, and A. Vadgama. Warpp: a toolkit for simulating high-performance parallel scientific codes. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, Simutools '09, pages 19:1–19:10, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
6. John Hennessy and David Patterson. *Computer Architecture - A Quantitative Approach*. Morgan Kaufmann, 2003.
7. Curtis L. Janssen, Helgi Adalsteinsson, Scott Cranford, Joseph P. Kenny, Ali Pinar, David A. Evensky, and Jackson Mayo. A simulator for large-scale parallel architectures. *International Journal of Parallel and Distributed Systems*, 1(2):57–73, 2010.
8. Hyun-Wook Jin, S. Sur, L. Chai, and D.K. Panda. Lightweight kernel-level primitives for high-performance mpi intra-node communication over multi-core systems. In *Cluster Computing, 2007 IEEE International Conference on*, pages 446 –451, sept. 2007.
9. Philip J. Mucci, Shirley Browne, Christine Deane, and George Ho. Papi: A portable interface to hardware performance counters. In *In Proceedings of the Department of Defense HPCMP Users Group Conference*, pages 7–10, 1999.
10. Ryutaro Susukita, Hisashige Ando, Mutsumi Aoyagi, Hiroaki Honda, Yuichi Inadomi, Koji Inoue, Shigeru Ishizuki, Yasunori Kimura, Hidemi Komatsu, Motoyoshi Kurokawa, Kazuaki J. Murakami, Hidetomo Shibamura, Shuji Yamamura, and Yunqing Yu. Performance prediction of large-scale parallell system and application using macro-level simulation. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08, pages 20:1–20:9, Piscataway, NJ, USA, 2008. IEEE Press.

Search-Based Ambiguity Detection in Context-Free Grammars

Naveneetha Vasudevan[1] and Laurence Tratt[2]

1 **Informatics, King's College London**
Strand, London, WC2R 2LS, United Kingdom. naveneetha@yahoo.com
2 **Informatics, King's College London**
Strand, London, WC2R 2LS, United Kingdom. laurie@tratt.net

Abstract

Context Free Grammars (CFGs) can be ambiguous, allowing inputs to be parsed in more than one way, something that is undesirable for uses such as programming languages. However, statically detecting ambiguity is undecidable. Though approximation techniques have had some success in uncovering ambiguity, they can struggle when the ambiguous subset of the grammar is large. In this paper, we describe a simple search-based technique which appears to have a better success rate in such cases.

1998 ACM Subject Classification F.4.2 Grammars and Other Rewriting Systems

Keywords and phrases Ambiguity, Parsing

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.142

1 Introduction

Context Free Grammars (CFGs) are widely used for describing formal languages, including programming languages. The full class of CFGs includes ambiguous grammars—those which can parse inputs in more than one way. Since this causes conceptual and performance problems, most parsing algorithms can parse only a narrow subset of CFGs, avoiding ambiguity issues altogether. However, this is not without cost: the subsets are restrictive and rule out useful actions such as composing grammars. The starting point for this paper is that parsing using the full class of CFGs is a useful activity.

Ambiguity is a huge problem for machine processed languages, such as programming languages. If an input can be parsed in two ways, which should be taken? Unfortunately, we know that it is impossible to statically detect whether an arbitrary CFG is ambiguous or not [6].

Over the years, therefore, there has been a steady stream of work trying to uncover ambiguity in arbitrary CFGs. Exhaustive methods such as AMBER [9] systematically generate strings to uncover ambiguity, but even medium sized grammars quickly lead to unmanageable huge state spaces. Approximation techniques, on the other hand, sacrifice accuracy for termination. For instance, ACLA [5] is an approximation method where the original language of the grammar is extended into an approximated language that can be expressed with a regular grammar. Since all the strings from the original language are also included in the approximated one, there are no false negatives reported. However, the approximated language may contain strings that may not be part of the original one, and therefore the method can report false positives. Noncanonical Unambiguity (NU) Test is another approximation technique, where the original grammar is converted to a bracketed grammar by adding two terminals – a derivation (d_i) and a reduction (r_i), where i is the

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 142–148

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

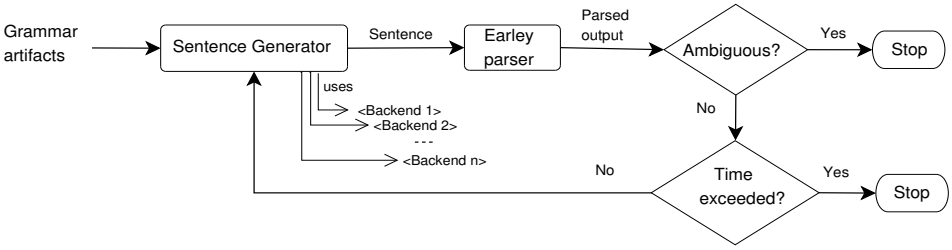


Figure 1 SinBAD architecture.

number of the production – at the front, and at the end of every grammar rule respectively. The introduction of these two terminals makes the bracketed grammar unambiguous. The challenge then, is to find two bracketed strings from the approximated grammar that map to a string in the original grammar. However, this method does not scale well for large grammars [3].

Hybrid approaches – where an approximation method is combined with an exhaustive method – increase the chances of detecting ambiguity. Basten's hybrid approach [4] – based on grammar filtering – applies an approximation method (NU Test) to filter out the unambiguous portions of the grammar, and then runs AMBER on the resulting smaller grammar to detect ambiguities. In principle, Basten's approach can be extended to other tools: ACLA, an approximation method, can be combined with CFG Analyzer [1], an exhaustive method, to search for ambiguous strings of bounded length. However, such hybrid approaches still rely on an exhaustive search although on a relatively smaller state space.

This paper is the first to explore a random search-based approach to grammar ambiguity detection. Given a grammar, our approach generates random strings, which are then parsed to detect ambiguity. In section 2.1 we describe our prototype tool: Search-Based Ambiguity Detection (SinBAD). In section 3 we set out the objective of our experiment, and then explain the choice of various data sets used for our experiment. In section 4 we compare and analyse our results. In section 5 we highlight the threat to validity of our random grammar generator, and finally in section 6 we conclude our experiment and provide future directions of our work.

2 Search-based ambiguity detection

Search-based techniques seek to find 'adequately' optimal solutions for problems that have no algorithmic solution and whose search space is too big to exhaustively scan. Such techniques have been applied to a wide range of problems including software itself (see e.g. [7]). Search-based techniques are either purely random or metaheuristic (such as hill climbing and genetic algorithms). Whereas in a random search the search space of candidate solutions is scanned randomly, in a metaheuristic search, a *fitness function* – to distinguish between a good and a poor solution – is used to guide the search. Since, this is the first paper to explore search-based techniques to ambiguity detection in CFGs, we have chosen the simplest search-based technique – a pure random search – for our experiment.

2.1 SinBAD framework

In this paper, we apply search-based techniques to ambiguity detection. We do so using a new tool, SinBAD, which allows us to experiment with different search-based approaches. Figure 1 shows SinBAD's architecture. Given a grammar and a lexer, the *Sentence Generator*

component generates random sentences using a *backend* instance. A backend, in essence, is an algorithm that governs how sentences are generated. For instance, a backend can use a unique scoring mechanism to favour an alternative when expanding a nonterminal, or one that can generate sentences of bounded length. The generated sentence is then fed to an Earley-based parser to check for ambiguity. The search stops when an ambiguity is found or when a time limit is exceeded. SinBAD can be downloaded from https://github.com/nvasudevan/sinbad.

2.2 Definition and Notations

A CFG is a four-tuple $\langle N,T,P,S\rangle$ where N is the set of nonterminals, T is the set of terminals, P is the set of production rules over $N \times N \cup T$ and S is the start symbol of the grammar. V is defined as $N \cup T$. A production rule $A: \alpha$ is denoted as $P[A]$ where $A \in N$, and α is V^*. We define a sentence of a grammar as a string over T^*. For a rule $P[A]$, $P[A]_{\text{alt}}$ denotes an alternative, and $\Sigma P[A]_{\text{alt}}$ denotes all its alternatives. The number of alternatives for a rule and the number of tokens in a rule are denoted as $\mathbb{N}(P[A])$ and $\mathbb{N}(P[A]_{\text{alt}})$ respectively. Notation $\mathbb{R}(L,n)$ indicates n items chosen randomly from a list L, and $\mathbb{R}[m..n]$ indicates a number chosen randomly between m and n.

2.3 Search-based backends

Given a grammar, Algorithm 1 describes how a sentence is generated. The function START is initialised with a grammar (G), the start time (t_s), the time duration (T) of search, and the threshold depth (D). To generate a sentence, we start deriving the start symbol S of the grammar by invoking the function GENERATE-SENTENCE recursively. To derive a nonterminal we randomly select one of its alternatives (line 11). We keep a note of when we have entered a rule and when we have exited. When the depth of the recursion exceeds a certain threshold depth, we start favouring alternatives (lines 8,9).

Algorithm 2 shows how an alternative is favoured for the Dynamic1 backend. When invoked for a rule, the function FAVOUR-ALTERNATIVE uses a scoring mechanism to favour an alternative. The score for an alternative is calculated as follows: terminal symbols are given a score of zero; for nonterminal symbols, the score is based on the ratio of their number of derivations that haven't been fully derived yet to the total number of derivations (line 8). One of the alternatives with a minimum score is then favoured.

3 Experiment

The objective of our experiment is to understand how well our search-based approach uncovers ambiguity. Since ambiguity is inherently undecidable, it is impossible to evaluate such a tool in an absolute sense. Instead, we evaluate our approach against two other tools – ACLA and AmbiDexter [2] – and on two sets of grammars: 1000 grammars that we have randomly generated[1]; grammars for Pascal, SQL, Java and C that have been manually altered to be ambiguous[2].

The three tools differ in their approach: ACLA uses an approximation technique; AmbiDexter uses a hybrid approach; and SinBAD uses a search-based approach. We evaluate these three tools for both sets of grammars for varying time limits – 10, 30, 60, and 90 seconds – to understand how long each tool takes to uncover reasonable quality results. For

[1] Available at https://github.com/nvasudevan/sinbad/tree/master/experiment.
[2] Taken directly from [4].

Algorithm 1 Algorithm for generating a sentence

```
1: function START(G, t_s, T, D)
2:     return GENERATE-SENTENCE(P[S], G, t_s, T, d = 0, D)
3: end function
4: function GENERATE-SENTENCE(P[A], G, t_s, T, d, D)
5:     exit if time_elapsed(t_s, T)
6:     Sen ← empty string
7:     P[A].entered ← P[A].entered + 1                    ▷ We enter rule
8:     if d ≥ D then
9:         P[A]_alt ← FAVOUR-ALTERNATIVE(P[A], G)
10:    else
11:        P[A]_alt ← ℝ(ΣP[A]_alt, 1)
12:    end if
13:    for each V ∈ P[A]_alt do
14:        if V ∈ N then
15:            Sen ← Sen + GENERATE-SENTENCE(P[V], G, t, T, d + 1, D)
16:        else
17:            Sen ← Sen + V
18:        end if
19:    end for
20:    P[A].exited ← P[A].exited + 1                      ▷ We exit rule
21:    d ← d − 1
22:    return Sen
23: end function
```

Algorithm 2 Algorithm for favouring an alternative for Dynamic1 backend

```
1: function FAVOUR-ALTERNATIVE(P[A], G)
2:     scores ← [ ]
3:     for each P[A]_alt ∈ ΣP[A]_alt do
4:         score_alt ← 0
5:         for each V ∈ P[A]_alt do
6:             if V ∈ N then
7:                 if P[V].entered > 0 then
8:                     score_alt ← score_alt + (1 − (P[V].exited/P[V].entered))
9:                 end if
10:            end if
11:        end for
12:        scores ← score_alt
13:    end for
14:    alts_min ← { alt | ∀alt ∈ ΣP[A] ∧ score_alt = min(scores) }
15:    return ℝ(alts_min, 1)
16: end function
```

the (generally much larger) programming language grammars, we also evaluate the tools for extended periods (180 and 300 seconds) as the number of production rules is much higher than for our random grammars.

We evaluate AmbiDexter for two versions of a grammar—unfiltered and filtered (with

SLR1). AmbiDexter provides an option for generating filtered versions of a grammar. For random grammars, we generate the filtered version, and for the altered programming language grammars, we take it directly from [4]. We evaluate SinBAD with the Dynamic1 and Dynamic2 backends for two threshold depths (D), 10 and 30. We have chosen these two values for depth to uncover reasonably long ambiguous fragments. Our experiment was performed on an Intel Core2 Quad Q9450 2.66GHz machine with 4 GB of memory. The maximum JVM heap size for ACLA and AmbiDexter was 2048Mb.

3.1 Random grammar generation algorithm

Algorithm 3 outlines the algorithm for our random grammar generator. We initialise nonterminal and terminal sets with equal numbers of symbols. To generate an alternative, a token is picked randomly from set V. Each rule can have 1 or more alternatives, and each alternative can have 0 or more symbols. The maximum number of alternatives for a rule and the maximum number of tokens in an alternative is controlled by the MAX_{alts} and $\text{MAX}_{\text{tokens}}$ parameters respectively. The MAX_{ϵ} controls the maximum number of empty alternatives.

Algorithm 3 An algorithm for generating a random grammar

```
1:  function GENERATE-GRAMMAR(MAX_alts, MAX_tokens, MAX_ε)
2:      P ← { }
3:      N ← Set of nonterminals
4:      T ← Set of terminals
5:      V ← N ∪ T
6:      N_ε ← ℝ(N, MAX_ε)
7:      for each A ∈ (N ∪ S) do
8:          N_alts ← ℝ[1..MAX_alts]
9:          while ℕ(P[A]) < N_alts do
10:             P[A]_alt ← [ ]
11:             N_tokens ← ℝ[1..MAX_tokens]
12:             while ℕ(P[A]_alt) < N_tokens do
13:                 P[A]_alt ← P[A]_alt + ℝ(V, 1)
14:             end while
15:         end while
16:         P[A] ← P[A] + [ ] if A ∈ N_ε          ▷ Append an empty list
17:     end for
18:     return ⟨N,T,P,S⟩
19: end function
```

All the grammars the algorithm generates are syntactically valid, though there is no guarantee that they resemble 'real-world' grammars. For example: a grammar with a start rule `S: x` can't be derived further; a rule `A: A` with no other alternatives never terminates.

4 Comparison and Analysis

Table 1 displays the results of our experiment. We now present a brief analysis of some of the most interesting parts.

Given a grammar, ACLA will report it to be ambiguous, unambiguous, or possibly ambiguous (that is, it is unsure if the grammar is ambiguous). For both sets of grammars,

Table 1 Number of ambiguities detected for random and programming language grammars.

		ACLA	AmbiDexter		SinBAD			
	Time	-	-		Dynamic1		Dynamic2	
	(seconds)		Unfiltered	SLR1	D=10	D=30	D=10	D=30
Random CFGs	10	81	355	356	357	15	499	26
	30	201	373	371	499	57	634	55
	60	316	376	371	545	54	631	80
	90	360	378	376	554	72	629	82
Altered real-world CFGs	10	14^{bc}	16^{ab}	16^{ab}	20	18^{b}	16^{ac}	17^{ab}
	30	14^{bc}	16^{ab}	16^{ab}	20	18^{b}	16^{ac}	18^{ab}
	60	14^{bc}	16^{ab}	16^{ab}	20	18^{b}	16^{ac}	18^{ab}
	90	14^{bc}	16^{ab}	16^{ab}	20	19^{a}	16^{ac}	19^{a}
	180	15^{bc}	18^{ab}	19^{b}	20	20	16^{ac}	19^{a}
	300	15^{bc}	18^{ab}	19^{b}	20	20	16^{ac}	20

a) Ambiguity not found for at least one of: Java.1, Java.3, and Java.4
b) Ambiguity not found for at least one of: C.1, C.2, C.4, C.5
c) Ambiguity not found for at least one of: Pascal.3, Pascal.5

ACLA performs better when we increase the time limit. For random grammars, ACLA did not report any grammar to be unambiguous. For the altered programming language grammars, Pascal.3 and Pascal.5 were reported to be possibly ambiguous. Analysis for the (large) C grammars – C.1, C.2 and C.4 – did not complete within a time limit of 300 seconds.

AmbiDexter fared better than ACLA for both sets of grammars. For random grammars, increasing the time limit does not lead to a significant increase in the number of ambiguities found. This is because AmbiDexter searches for ambiguity based on increasing sentence length. Therefore, for grammars with a short ambiguous fragment, AmbiDexter is quick to find it. However, when the ambiguous fragment is long, AmbiDexter struggles. For the altered programming language grammars, the results were slightly better for the filtered version set. This is because in filtered grammars, production rules that do not contribute to ambiguity are filtered out, thus resulting in a smaller state space. Further, we noted that for larger grammars (such as C), increasing the time limit lead to better results.

SinBAD, for random grammars, performs better for a lower value of threshold depth (D=10) than for a higher value (D=30). This is because, for case D=10, sentence generation is quick whereas for case D=30, sentence generation takes much longer. Generating sentences quicker allows the search to try a greater number of sentences possible, thereby increasing the chances of detecting ambiguity. Further, Dynamic2 – which has a better mechanism to converge sentence generation than Dynamic1 – performs better. For the altered programming language grammars, Dynamic1 performs better than Dynamic2. Dynamic1 uses a scoring mechanism that ensures every alternative gets an opportunity to be selected for sentence generation. Dynamic2, however, uses a scoring mechanism that focuses on converging the sentence generation. As a result, Dynamic1 covers a much wider area of the search space than Dynamic2. As table 1 shows, SinBAD performs much better on random grammars than the other tools, and performs at least as well on altered programming language grammars.

We also noted that whilst the number of ambiguities found for ACLA, AmbiDexter, and SinBAD's Dynamic1 stayed the same or increased, Dynamic2 got slightly worse with increased time limits and D=30. This is because both ACLA and AmbiDexter search through the state space systematically, and therefore the search space for higher time limits is inclusive of the search space for lower time limits. SinBAD, however, randomly selects points in the search space, and can give substantially different results from run to run.

5 Threats to validity

The most obvious threat to validity is our random grammar generator. We have no easy way of being confident that the CFGs it produces span the entire possible set of CFGs. Although we wrote the generator without any particular ambiguity tool in mind, it may produce a subset of CFGs which unintentionally favour SinBAD's algorithms. In the future, we hope that a CFG equivalent of the work on random generation of automata [8] may be developed. By using Basten's set of manually altered real programming language grammars, we have some confidence that SinBAD's algorithms work well beyond our random grammars.

6 Conclusions

In this paper, we introduced the concept of a search-based approach to CFG ambiguity detection. Our experiments show that simple techniques give promising results, detecting a larger number of ambiguities in random grammars than previous tools, and executing in reasonable time. Our next step is to add more tools to the study and perform a larger experiment with more real-world-esque grammars to see if these initial results apply to the sort of CFGs that tend be to be used in practice.

References

1 Roland Axelsson, Keijo Heljanko, and Martin Lange. Analyzing context-free grammars using an incremental sat solver. In *Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part II*, ICALP '08, pages 410–422. Springer-Verlag, 2008.

2 Bas Basten and Tijs van der Storm. Ambidexter: Practical ambiguity detection. In *Tenth IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2010, Timisoara, Romania, 12-13 September 2010*, pages 101–102. IEEE Computer Society, 2010.

3 H.J.S. Basten. Msc. thesis. Master's thesis, 2007.

4 H.J.S. Basten and J. J. Vinju. Faster ambiguity detection by grammar filtering. In *Proc. of the Tenth Workshop on Language Descriptions, Tools and Applications*, pages 5:1–5:9. ACM, 2010.

5 Claus Brabrand, Robert Giegerich, and Anders Møller. Analyzing ambiguity of context-free grammars. *Science of Computer Programming*, 75(3):176–191, March 2010.

6 David G. Cantor. On the ambiguity problem of backus systems. page 477–479, 1962.

7 Mark Harman. The current state and future of search based software engineering. In *FOSE*, pages 342–357, 2007.

8 Pierre-Cyrille Héam, Cyril Nicaud, and Sylvain Schmitz. Random generation of deterministic tree (walking) automata. In *Proceedings of the 14th International Conference on Implementation and Application of Automata (CIAA'09)*, volume 5642 of *Lecture Notes in Computer Science*, pages 115–124. Springer-Verlag, July 2009.

9 Friedrich Wilhelm Schröer. Amber, an ambiguity checker for context-free grammars. Technical report, 2001. `http://accent.compilertools.net/Amber.html`.

Introduction to Team Disruption Mechanisms

Andrada Voinitchi, Elizabeth Black, and Michael Luck

Department of Informatics, King's College London, UK
{andrada.voinitchi,elizabeth.black,michael.luck}@kcl.ac.uk

Abstract

This paper discusses how teams can be disrupted. More specifically, it discusses the steps that need to be taken in order to fully understand team disruption and design efficient mechanisms to disrupt teams. In order to answer the high-level question of how to disrupt teams, a few other questions need to be tackled first: what is a disrupted team? What are the crucial elements that make a collection of agents function as a team? Can norms, incentives or other mechanisms be used to disrupt these elements? How would we evaluate their efficiency? We first present the ideas of team and team disruption and motivate the need for these concepts to be properly defined. Secondly, we introduce an idea of team-disruption mechanism that we will further investigate. Lastly, we provide a long-term perspective and identify contributions that our research will make in the multi-agents field.

1998 ACM Subject Classification I.2. Computing Methodologies Artificial Intelligence

Keywords and phrases Team disruption, multi-agent systems, organisations, teams, goals

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.149

1 Introduction

In order to better expose and motivate the question of how teams can be disrupted we consider a real-life scenario: a team of five terrorists are planning to place a bomb in a tube station. The bomb is heavy and needs to be smuggled in, part by part, in order for the station staff not to get suspicious. This can be done over time while keeping the parts hidden in the tube station or it can be done on the same day. Furthermore, the terrorists need to coordinate in order to assemble the bomb on the premises and leave the station before the bomb is detonated. What the terrorists do not know is that another team of undercover agents from the secret services have found out about their plan. We can say that the two teams compete, in the sense that only one of the teams can be successful in achieving its goal at a certain point in time:

- the terrorists want to blow up the station and bring about a state of the world where the specific station is destroyed;
- the secret services want to preserve the station and keep a state of the world where the specific station is intact.

There are two questions (of particular interest to us) arising from this example: how can a member of the secret services team infiltrate the terrorist team and disrupt their activity and, based on information provided by the secret services, how can the government introduce regulations in order to make it more difficult for terrorists to put bombs in tube stations?

This can also be applied in computational systems where teams of computational agents compete. It provides an insight into how team activity can be disrupted by a computational agent belonging to a competing team or by a legislator (the legislator is an agent that has the capability to introduce norms, which are rules that govern the behaviour of a computational

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 149–155

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

system). These two perspectives constitute a starting point for the process of designing and analysing team disruption mechanisms.

Team disruption is useful in various ways: when two teams compete, one of the teams may be at an advantage if they have mechanisms that can disrupt the other team's activity. Furthermore, team disruption is also useful when thinking about teams of humans: looking at our earlier example we can tell that a team disruption mechanism would be a good tool for the secret services to prevent a terrorist attack. Lastly, when it comes to team design, we need to ensure that we design our teams to be as efficient as possible. Knowing what can disrupt a team's activity we also know what to account for in the design, in order to make our teams more robust.

The contributions of this paper are two-fold: first, we provide a definition of team disruption; second, we suggest possible team disruption mechanisms that are to be investigated in future research.

The team disruption question discussed here is split into a set of preliminary questions that are discussed in detail throughout the paper. Section 2 focuses on a discussion of work that is relevant to this research such as a definition of teamwork and existing teamwork theories. Section 3 provides a breakdown of the team disruption question as well as a few ideas towards a methodology to be used in order to achieve team disruption. Section 4 comments on future work and its implications.

2 Related Work

While there has been much research conducted on teams and teamwork, as we discuss below, there is nothing specifically addressing the issue of team disruption.

Teamwork is a cooperative effort by the members of a team to achieve a common goal [9, 10]. It involves cooperative behaviour, coordination and at least one common goal among agents in the team. A team is formed by agents that agree to work together towards achieving a goal. However, between joining a team and achieving a goal there is a gap that needs to be resolved: how can agents work together in order to achieve the goal? Teamwork theories such as the Joint Intentions Theory [7], Joint Responsibility Theory [5, 6] and the Shared Plans Theory [3, 2] provide some answers to this question, describing mechanisms for achieving teamwork among agents. They are very relevant to our work because we need to consider how teamwork happens in order to be able to think about disrupting it.

3 Team disruption

This section is concerned with issues of what makes a team function and how it can be disrupted. In order to provide a better understanding of the technical concepts involved, preliminary definitions are provided.

3.1 Preliminary definitions

Agent: an agent is an autonomous entity that is characterised by its goal-directed behaviour, reactiveness to changes in its environment and its social ability (it is able to communicate with other agents). Two or more agents that interact within an environment form a *multi-agent system* (MAS). Furthermore, within a MAS, agents can work together towards achieving a common goal. This is referred to as a *team*: a team is a set of agents that share a common goal and cooperate in order to achieve it. This definition is further discussed in Section 3.2. In a MAS, agents may behave according to certain rules. These rules are called *norms*. In

real life, norms are rules that govern a society and are used in order to regulate the behaviour of its individuals [8]. Norms in a MAS are the equivalent of written laws in a human society: agents need to comply with them in order to be rewarded or in order to prevent being punished. There are three kinds of norms: obligations, permissions and prohibitions [8] and only specific agents are allowed to introduce norms in a team. A MAS that supports the use of norms is also called a *normative MAS*.

3.2 What makes a team function?

A team is a set of agents that share a common goal. In a multi-agent setting, teamwork involves cooperative behaviour, coordination and communication [4]. Furthermore, teams are dynamic as agents can join or leave at any point in time.

Returning to the example mentioned in the introduction, the team of terrorists is a team because all of its members have one goal in common: destroy the tube station by detonating a bomb. Furthermore each of the terrorists has a role in the team: one terrorist is in charge of managing the others, some terrorists are in charge of carrying the bomb parts and so on.

It can be inferred from the previous definition of teamwork that, in order to function, a team has to support the following aspects: cooperation (the agents in a team need to act towards achieving the team goal instead of prioritizing their individual goals), coordination (to avoid redundancy: for example, to ensure that no two members attempt the same task simultaneously), communication (agents need to be able to communicate, as they need to coordinate and avoid duplicating work) and a common goal (if there is no common goal among all of the agents in a team then there is no reason for teamwork).

The aspects mentioned above are a few examples of what makes a team function. This remains a crucial question that has to be further researched in order to be able to identify what aspects can be sabotaged in order to disrupt a team.

3.3 What is a disrupted team?

The notion of a *disrupted team* is crucial for achieving an understanding of how teams can be disrupted. Team disruption, as we envision it, is based on the idea of rendering the team goal unachievable or causing a major delay in the achievement of the team goal (under the assumption that the team goal is initially achievable in a timely fashion). For example, in the context of our scenario, if one of the terrorists is observed carrying something suspicious by an undercover agent from the secret services team, then the terrorist has to hide, delaying the bombing, hence the team experiences disruption in the form of a delayed achievement of the team goal. If caught, the carrier terrorist is no longer able to deliver its part of the bomb, hence the goal is compromised and no longer achievable.

An understanding of what team disruption involves also relies on the way one thinks about reducing team efficiency. Is it enough to think of a disrupted team based on a delay in achieving the team goal or rendering the goal unachievable? How does this relate to lessening the number of members in the team? Does it imply disabling key members? It can also be assumed that finding ways of hindering any of the four aspects of a functioning team (cooperation, coordination, communication and common goal) can disrupt its activity.

A relevant aspect of team disruption is the agent that disrupts the team: a team can be disrupted by a malicious agent (possibly coming from another team) that has infiltrated within the team or through new regulations imposed by a legislator in the community that the team operates in.

When considering team disruption, we can consider not just whether it succeeds or fails but also the degree of success or failure, providing an idea of the extent to which a team can be disrupted. The concept of *magnitude of disruption* must be introduced when thinking about a disrupted team. This idea is at a very early stage and will be used in the future in order to evaluate team disruption mechanisms, thus it will not be discussed further here.

3.4 How to disrupt a team?

There are a few mechanisms that have the potential to disrupt teams (e.g. introducing norms, providing agents with incentives [1] to leave their team, reducing resources available to a team and so on). Here we will only address introducing norms in more detail, because of space restrictions.

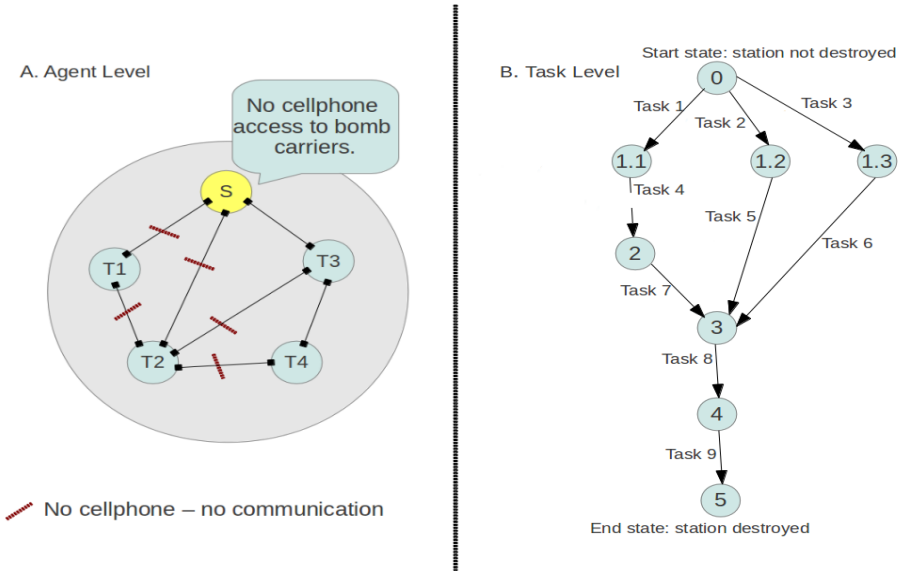


Figure 1 Disruption scenarios using norms: two perspectives for disrupting teams are illustrated both at agent level and at task level. In Figure 1.A. the nodes represent agents that are part of a terrorist team, while the links represent communication between agents. Agents T1-4 are members of the terrorist team while agent S is a member of the secret services that has infiltrated in the team. Figure 1.B. shows how the terrorist team goal is represented at task level. The nodes in the graph represent states of the world that the team operates in and the edges represent tasks that are accomplished in order to transition from a state to the other.

When investigating team disruption, we can look at two levels of the team: the task level and the agent level (Figure 1). The agent level models the agent connections using a graph. The connections represent the ability of agents to communicate. The task level provides more detail on the sequence of steps towards achieving the team goal (task).

For example in Figure 1.A, agent S is an undercover agent infiltrating the terrorist team. In order to infiltrate the terrorist team, agent S declares to agent T3 (we assume agent T3 is in charge of managing the team) that it shares the goal of bombing a tube station. Once it joins the team, agent S gains the authority of setting norms for the team. It can then introduce a norm prohibiting agents that carry bomb parts (namely T1 and T2) from

carrying cellphones, hence breaking their communication. This will make it more difficult for them to meet up and assemble the bomb. With no communication, agents T1 and T2 may not be able to find a common meeting point, leading to the goal being compromised. In the case where they meet accidentally, this may still delay the achievement of the goal as they may not meet in the place where they are supposed to assemble the bomb. This example illustrates team disruption at the agent level. From here we can infer that a team may be disrupted at the agent level by the introduction of norms that affect communication between agents that fulfill certain roles.

In Figure 1B, the same team of terrorists wishes to bomb a tube station. The topology of the team is as illustrated in Figure 1A. However, in this scenario, we do not focus on hindering communication for disrupting the team. This scenario relies on the fact that the team has a plan on how the bombing will be carried out. As shown in Figure 1B, the plan is composed of prioritized states that need to be reached in a specific order such that a connection between the start and the goal states is provided, as represented in a goal graph. In some cases, there may be more than one way of reaching the goal. The transition between states is realized through agent actions (tasks). Here, the states are represented as follows (we assume that the bomb is composed of three parts).

- 1.1: First bomb part is in the station.
- 1.2: Third bomb part is in the station.
- 1.3: Second bomb part is in the station.
- 2: Second bomb part is in the station.
- 3: All bomb parts are in the station.
- 4: Bomb is assembled (and in the station).

Tasks represent sets of actions performed by agents, as specified.

- Task 1: agent T1 carries first bomb part in the station.
- Task 2: agent T1 carries third bomb part in the station.
- Task 3: agent T1 carries second bomb part in the station.
- Task 4: agent T2 carries second bomb part in the station.
- Task 5: agent T2 and T3 carry the first and second bomb parts in the station.
- Task 6: agent T2 and T3 carry the first and third bomb parts in the station.
- Task 7: agent T3 carries the third bomb part into the station.
- Task 8: agent T2 assembles the bomb.
- Task 9: agent T4 detonates the bomb.

As shown in the example in Figure 1B, more members of the team can either carry the bomb parts simultaneously, meet in the station and assemble the bomb (nodes 1.2 and 1.3 of the goal graph), or one member can carry all of the bomb parts, part by part, store them in the station and assemble it when all of the parts have arrived (node 1.1. of the goal graph). This example is different from the previous one (Figure 1A) in that it does not imply direct sabotage by an agent infiltrating. Rather, agents from a competing team can use their observations to convince legislators to introduce norms: if it is known that bomb parts can be stored unattended in tube stations, the legislators (i.e. government) can introduce a law (norm) stating that no bags can be left unattended in the tube station. Unattended bags are seized and destroyed. Because State 2 from the goal graph is no longer achievable, the team may be delayed in achieving the goal (bombing) because their options were reduced. Furthermore, if legislators introduce a norm restricting bag size for travelers, larger parts of the bomb may not be introduced into the station at any time, thus the bombing is fully compromised.

When considering disruption at task level, a few issues need to be considered for an efficient approach, as follows.

- First, in order to render the goal unachievable, all paths to the goal need to be compromised (in the goal graph).
- Second, some tasks may be crucial to accomplishing the team goal: if these tasks are not accomplished, the goal is rendered unachievable (such as Task 8 in the example in Figure 1B). These are considered *critical tasks* and the agents that can perform them are deemed critical agents.

4 Conclusion and future work

Based on the results that will be obtained from the planned simulation of all of the mechanisms presented, a further step is to be taken, comparing the outcomes of the scenarios based on criteria such as the time it takes a team to achieve a goal and if the team can achieve its pre-disruption goals. A further investigation will establish whether each of these mechanisms can work individually or whether applied together they will have a greater impact.

If it is demonstrated that any of the methods described here can be used in order to disrupt team activity further questions need to be answered. For example, if introducing norms can be responsible for team disruption, a number of issues need to be further investigated: how would such norms be generated? Who would introduce and enforce such norms? What proportion of the agents in a team would need to comply with such norms in order for the team to be disrupted? If all of the members of a team need to comply with the norms, who would supervise the norm enforcers in order to assure compliance? As a further step, it is vital to compare all the successful approaches and provide a detailed analysis on which of these approaches work better and under which circumstances.

References

1 Y. Chen, J. Kung, D.C. Parkes, A.D. Procaccia, and H. Zhang. Incentive design for adaptive agents. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '11, pages 627–634, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems.

2 B.J. Grosz, L. Hunsberger, and S. Kraus. Planning and acting together. *AI Magazine*, (20):23–34, 1999.

3 B.J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, October 1996.

4 B. Horling and V. Lesser. A survey of multi-agent organizational paradigms. *Knowledge Engineering Review*, 19(4):281–316, December 2004.

5 N.R. Jennings. On being responsible. *SIGOIS*, 13(3):8–, December 1992.

6 N.R. Jennings. Commitments and conventions: The foundation of coordination in multiagent systems. *Knowledge Engineering Review*, 3(8):223–250, 1993.

7 H.J. Levesque, P.R. Cohen, and J.H.T. Nunes. On acting together. In *The 8th National conference on Artificial intelligence - Volume 1*, AAAI'90, pages 94–99. AAAI Press, 1990.

8 S. Modgil, N. Faci, F. Meneguzzi, N. Oren, S. Miles, and M. Luck. A framework for monitoring agent-based normative systems. In *The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '09, pages 153–160, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.

9 D.V. Pynadath and M. Tambe. Multiagent teamwork: analyzing the optimality and complexity of key theories and models. In *The 1st international joint conference on Autonomous*

agents and multiagent systems: part 2, AAMAS '02, pages 873–880, New York, NY, USA, 2002. ACM.

10 M. Tambe. Towards flexible teamwork. *Journal of Artificial Intellience Research*, 7(1):83–124, September 1997.

Self-Learning Genetic Algorithm For Constrains Satisfaction Problems*

Hu Xu[1] and Karen Petrie[2]

1 **Computing School,**
QMB 1.10, University of Dundee
huxu@computing.dundee.ac.uk
2 **Computing School,**
QMB 2.10, University of Dundee
karenpetrie@computing.dundee.ac.uk

Abstract

The efficient choice of a preprocessing level can reduce the search time of a constraint solver to find a solution to a constraint problem. Currently the parameters in constraint solver are often picked by hand by experts in the field. Genetic algorithms are a robust machine learning technology for problem optimization such as function optimization. Self-learning Genetic Algorithm are a strategy which suggests or predicts the suitable preprocessing method for large scale problems by learning from the same class of small scale problems. In this paper Self-learning Genetic Algorithms are used to create an automatic preprocessing selection mechanism for solving various constraint problems. The experiments in the paper are a proof of concept for the idea of combining genetic algorithm self-learning ability with constraint programming to aid in the parameter selection issue.

1998 ACM Subject Classification I.2.6 Learning

Keywords and phrases Self-learning Genetic Algorithm, Constraint Programming, Parameter Tuning

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.156

1 Introduction

The selection of a suitable preprocessing levels for a given constraint problem is an important part of constraint programming(CP). Efficiently tuning a constraint solver will shorten the search time and reduce the running cost. The key to increasing the search speed for a constraint solver is partially due to tuning the solvers parameters [9]. Currently the job of tuning the parameters is done by hand. The skilled researchers picks up the most suitable preprocessing method using previous experience from similar classes of problems. In most cases the best preprocessing method in similar classes of problems provide a useful clue to aid the researchers selection. However this learning curve could be a barrier to novice user in learning how to efficiently use a CP solver.

Genetic algorithms are a classic global optimization method posed by John Holland [7], which mimic the competition of organisms in nature and the mechanisms of evolution. Genetic algorithms are usually implemented in a computer simulation in which a population of abstract representations of candidate solutions to an optimization problem evolves toward better solutions. In the field of configuration tuning, Carlos [10] has posed a gender-based genetic

* This work was funded by Computing School of Dundee University and Henry Lester Turst

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 156–162

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

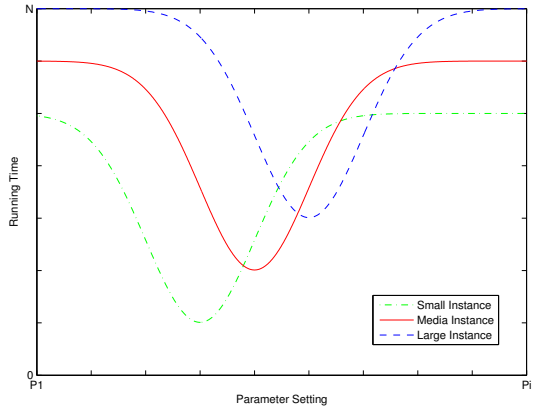


Figure 1 The Distribution of the Effect of Preprocessing for Various of Constrained Satisfaction Problems.

algorithm for the automatic configuration of algorithms. In this paper genetic algorithms are chosen to select preprocessing method for constraint satisfaction problems. There are two main reasons to choose genetic algorithms to optimize preprocessing selection. One is that genetic algorithm have a powerful ability to tackle optimization problems which lack auxiliary information [1]. Another is that genetic algorithm do parallel search rather than linear search [4]. Each chromosome races against another in each generation. Therefore the idea of combining genetic algorithm and constraint programming seems worth exploring. Automatic tuning will lead to improvements over manual tuning by researchers themselves. ParamILS and CALIBRA [8] have shown the efficiency and possibility of automatic configuration for constraints solver. However, the general framework of combing genetic algorithm and constraint programming and the exploration of parameter sensitivity of genetic algorithm to any problems, has not been achieved. Regrading this situation we proposed a genetic based automatic method [12] for tuning minion [3] (method refered to as GACM) which is one of the most efficient constraint solver in the world. In the constrained problem and their preprocessing obey the normal (or Gaussian) distribution [5]. In most time the distribution of the best preprocessing methods wouldn't changed or slightly changed in the same classes of the constraint satisfaction problems. Fig 1 shows that the best prepossessing could also gradually move with the increase of the scale of the constraint satisfaction problem. Therefore the best prepossessing method of a specific problem could learn from others in the same class of problems. Meanwhile the search ability of genetic algorithm can improve by narrow the starting population domain [4].Therefore this paper will propose a new self-learning mechanism which is based on a new starting population and our pervious work.

2 Self-Learning Genetic Algorithm

Before self-learning genetic algorithms, Standard genetic algorithms will be introduced. In Standard Genetic Algorithms, the starting population is randomly generated because the search domain is unknown and the random chromosomes keeps the variety of the population to prevent early convergency in evaluation. However if the search domain is limited to a specific area it will improve the search speed for evaluation. We can use this by creating a good starting popluation. The self-learning genetic algorithm is based on this idea. When we solve small scale constraint satisfaction problem it is easy to find the best or good

Algorithm 1 Self-Learning Genetic Algorithm

if $T(C_S) <$ Time limit **then** ▷ $T(C_S)$ is the running time of Solving some constrained satisfaction problems with small instance
  $P_L \leftarrow$ Best preprocessing for small instance ▷ P_L is the starting population for large instance
else
  $P_L \leftarrow$ Good preprocessing for small instance by standard genetic algorithm
end if
repeat
  $SGA(P_L)$ ▷ Using standard genetic algorithm to search better preprocessing with the starting population P_L
  $\lambda \leftarrow$ Best preprocessing for large instance by Standard Genetic algorithm ▷ λ is the current best preprocessing method found for optimization problem
until $\lambda =$ the best preprocessing or the searching time is out of time limit
return λ

preprocessing method within a acceptable running time. Those preprocessing methods will provide a cue for searching for a good preprocessing methods in large scale problems. Before the experiments the working principle of standard genetic algorithm for selecting preprocessing level will be introduced.

The first step of a genetic algorithm (GA) is called the encoding which is to construct the suitable chromosome for the optimization problem. Encoding in genetic algorithm is to transfer solutions of optimization problem to the chromosomes. Each chromosome presents one possible solution. The optimal or best solution will be gained by competing chromosomes. In our self-learning genetic algorithm, each preprocessing method was encoded as a chromosome.

Fitness describes the ability of an individual to reproduce in biology. The Fitness function is the function which evaluates the difference between the desired result and the actual result. In problem optimization, GA uses a fitness function to evaluate each individual and provide the information to the evolution.

The Selection in genetic algorithm is a strategy which allows the perfect parents (with high fitness) to have more of a chance to be selected to generate the next generation. In our genetic configurator, the selection is the roulette wheel selection. Roulette wheel selection is a way of choosing individuals from the population of chromosomes in a way that is proportional to their fitness. Roulette does not guarantee that the fittest member goes through to the next generation, merely that it has a very good chance of doing so.

Crossover can improve the whole population fitness quickly by mating parents to produce an offspring. It is a very important operator in genetic algorithms. Single point crossover is the basic and most common crossover in genetic algorithms because it can be easily understood and realized. Mutations which change one or more genes in an individual is another operator used in GA. Mutation can help genetic algorithms escape the local maximum state by creating a new gene string. As with crossover, mutation also has a mutation rate to control the amount of mutation in the recombination of each generation. The mutation rate is the probability of a mutation happen. According to the mutation rate, any bit in each chromosome has the chance to do a mutation.

Generally machine learning makes predictions by training, validation and testing itself existing data [11]. Self-learning genetic algorithm (refered to as SLGA) is the algorithm

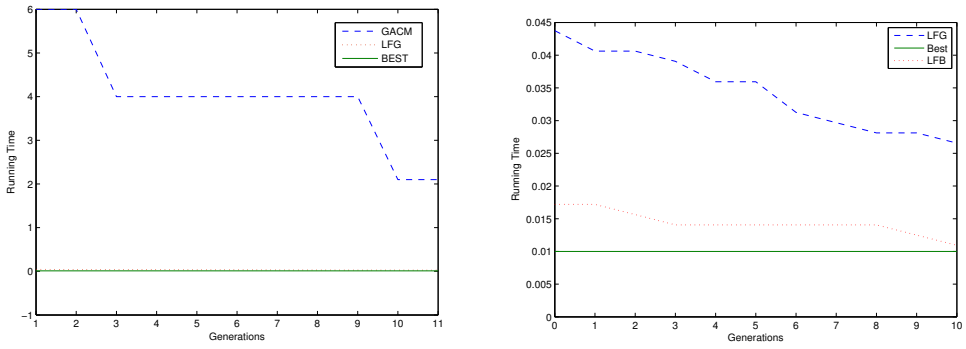


Figure 2 The Efficiency of Self-Learning Genetic Algorithm in Solving Lanford Number Problems. The X axis is the generation number of genetic algorithm to find the best preprocessing for Lanford Number problem. The Y axis is the running time for finding a solution of Lanford Number problem with relative preprocessing setting.The left graph shows the efficiency comparison between standard genetic algorithm and self-tranning genetic algorithm which learn the experience from the pervious evolutionary result for small instance. The right graph in Figure 2 shows the efficiency of two different strategies of self-learning genetic algorithm in solving the Landford number problem.

which help to make the preprocessing prediction by using the previous experience on the same classes of constraint satisfaction problem. Self-Learning genetic algorithm improve the search speed by defining the specific starting population instead of the normal random starting population. The starting population of a Self-Learning genetic algorithm is gained from the data training of the same class of small instance problems.There are two strategies to realize the Self-Learning mechanism. Learning From Best (refered to as LFB) strategy is to define the starting population with the best preprocessing which was gained by solving small instance with whole preprocessing possibility. Learning From Genetic algorithm (refered to as LFG) strategy is to define the staring population with the suggested preprocessing which was got by solving small instance with our pervious genetic method.

The pseudocode of self-learning genetic algorithm introduces SLGA's woking principle and the way of applying those two strategies for different problems. It shows that the self-learning genetic algorithm firstly evaluated the running time of Solving some constrained satisfaction problems with small instance. If it is possible to find the best processing for small instance problem, the starting population for large instance problems will be initialized with the best processing for small instance problems or else the optimal processing gained by GACM for small instance problems. According to the suggested starting population from pervious experience, the standard genetic algorithm will be applied to find the best or optimal processing for large instance problem. The standard genetic algorithm will explore better processing generation by generation. The evolutionary search loop will stop when he best preprocessing is found or the searching time is out of time expected.

3 Experiment Design

To prove the efficiency of the self-learning genetic algorithms, two different starting populations were chosen which were mentioned in the methodology part. One starting population is the top few of the best preprocessing of all the possible preprocessing combinations, another one is the top preprocessing gained from standard genetic algorithm. The efficiency of those two strategies (LFB and LFG) will compared with each other and with standard genetic

algorithm as well (GACM). In this paper the optimization problems chosen are the BIBD, the N-queen problem, Golomb and the Landford's number problem[1]. These four classical constraint problems were chosen as optimization problem for testing the self-learning genetic algorithm. The computational complexity of N-Queen problem depend on one variable. The complexity of Open Stack Problem is up to the instance provided and the complexity of Langfords Number Problem depends on multi-variables. From the definition description of problems, it shows that those four constraint problem are very different to each other . We hope that the self-learning genetic algorithm could be applicable to different constraint satisfaction problem. Following the David's MicroGA Settings [2], the crossover rate is 0.5 and the mutation rate is 0.04 in all experiments. Each trial was run 10 times and we observe the average of the minimums.

4 Experimental Results

Figure 2 shows the efficiency of self-learning genetic algorithm to solve the Landford problem. There are three curves in the left graph: Best, LFG and GACM. The best curves is the minimum running time for solving Landford number problem with best preprocessing. The GACM curve is the efficiency of using genetic algorithm to find better preprocessing for optimisation problems. The LFG curve shows the self-learning genetic algorithm that learn experience from pervious genetic algorithm evolution for the same class of problems. Its shows a standard genetic algorithm can gradually approach the best preprocessing methods after a few generations. But It clearly shows that the LFG can more easily and quickly approach the best result by inheriting the useful information from others similar small instances.

There are three curves in the right graph: Best, LFG and LFB. The best curves is the minimum running time same as in the left figure. The LFB curve shows the self-learning genetic algorithm that learn experience from the best processing for the same class problems. The LFG and the LFB curves both shows the efficiency of the self-learning genetic algorithm to search for the best preprocessing method. The LFB selects the best preprocessing setting of all possibility of small scale problem as the starting population. The LFG chooses good preprocessing methods as a starting population which is gained from solving small scale problems with a standard genetic algorithm. They both approach the best preprocessing setting step by step as we expected. Although the approach speed of LFG is faster than LFB, LFB still has better solutions due to the advantage in the starting population which we can find from the definition of LFB and LFG.

To convince the correction and efficiency of Self-tanning genetic algorithm for other problems, it was applied to solve the other three problems: BIBD[6], N-Queen problem and Golomb problem. In reality it is not always possible to gain all possibility of preprocessing combination from optimized problem which uses small instance due to the complexity of preprocessing. Therefore only the lFG strategy of self-learning genetic algorithm was applied to solve three optimization problems.

Table 1 describes the efficiency of the self-Learning genetic algorithm in solving different problems by comparing standard genetic algorithm. Each value in the table represents the running time of finding solution with the best found preprocessing. In all the optmization problems the LFG could find better solution than the standard genetic algorithm. Especially in Golomb problem the LFG could find the better solution but GA can't. It is obvious that the LFG has stronger ability than GA on searching for the best preprocessing method. The

[1] All from `http://www.csplib.org`

Table 1 The Efficiency of Self-Learning Genetic Algorithm in Solving Different Problems by comparing Standard Genetic Algorithm.

	BIBD	Langford	N-Queen	Golomb
GA	5.3 s	0.266 s	0.33 s	N/A
LFG	4.5 s	2.1 s	0.04 s	8.7 s
Best	3.2 s	0.01 s	0.04 s	6.6 s

curves in fig. 2 and table 1 shows that the self-learning genetic algorithm can quickly approach the best preprocessing within a few generations no matter which starting population strategy is chosen. The LFB is quicker than the learning LFG, but the approaching speed is slower. It means that the LFB strategy could be considered for self-learning genetic algorithm when the running time for small instance is small. When the searching time of optimized problem is unknown the LFG strategy is a better idea.

5 Future work

The results show the self-learning genetic algorithm are efficient methods on the preprocessing selection of solving constraint satisfaction problems . However there are a few challenges we need to face in the future. In this paper four classic problems were picked up to verify the efficiency of self-learning genetic algorithm on medium size scale problem. More and larger scale problems such as car sequence problem will be chosen to explore the efficiency and the limitation of self-learning genetic algorithm. Currently the best model to solve a constraint satisfaction problem is selected by hand by a researcher in the field. The next step is to apply self learning genetic algorithms to find the best model for a constraint satisfaction problem.

References

1 C. Ansótegui, M. Sellmann, and K. Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In *Principles and Practice of Constraint Programming-CP 2009: 15th International Conference, CP 2009 Lisbon, Portugal, September20-24, 2009 Proceedings*, page 142. Springer, 2009.

2 David L. Carroll. Chemical laser modeling with genetic algorithms. *AIAA Journal*, 34:338–346, 1996.

3 Ian P. Gent, Christopher Jefferson, and Ian Miguel. Minion: A fast scalable constraint solver. In *Proceedings of the 17th Eureopean Conference on Artificial Intelligence (ECAI'06)*, pages 98–102, 2006.

4 David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

5 Carla P. Gomes, Bart Selman, Nuno Crato, and Henry Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning*, 24:67–100, 2000. 10.1023/A:1006314320276.

6 Brahim Hnich, Zeynep Kiziltan, and Toby Walsh. Modelling a balanced academic curriculum problem. In *Proceedings of CP-AI-OR-2002*, pages 121–131, 2002.

7 John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* 1992.

8 Frank Hutter, Holger H. Hoos, and Thomas Stützle. Automatic algorithm configuration based on local search. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, AAAI'07, pages 1152–1157. AAAI Press, 2007.

9 Lars Kotthoff, Ian Miguel, and Peter Nightingale. Ensemble classification for constraint solver configuration. In *CP'10*, pages 321–329, 2010.

10 Tony Lambert, Carlos Castro, Eric Monfroy, María Riff, and Frédéric Saubion. *Hybridization of Genetic Algorithms and Constraint Propagation for the BACP*, volume 3668 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005.

11 Simon Rogers and Mark Girolami. *A First Course in Machine Learning*. Chapman & Hall/CRC, 1st edition, 2011.

12 Hu Xu, Karen Petire, and Keith Edwards. Genetic based automatic configuration for minion. *Doctoral Program at 2011 International Conference on Principles and Practice of Constraint Programming*, pages 91–96, 2011.