STAIRS 2014

proceedings of the 7th European Starting AI Researcher Symposium

Endriss, U.; Leite, J.

STAIRS 2014

Frontiers in Artificial Intelligence and Applications

FAIA covers all aspects of theoretical and applied artificial intelligence research in the form of monographs, doctoral dissertations, textbooks, handbooks and proceedings volumes. The FAIA series contains several sub-series, including "Information Modelling and Knowledge Bases" and "Knowledge-Based Intelligent Engineering Systems". It also includes the biennial ECAI, the European Conference on Artificial Intelligence, proceedings volumes, and other ECCAI – the European Coordinating Committee on Artificial Intelligence – sponsored publications. An editorial panel of internationally well-known scholars is appointed to provide a high quality selection.

Volume 264

Recently published in this series

Vol. 263. T. Schaub, G. Friedrich and B. O'Sullivan (Eds.), ECAI 2014 – 21st European Conference on Artificial Intelligence
Vol. 262. R. Neves-Silva, G.A. Tshirintzis, V. Uskov, R.J. Howlett and L.C. Jain (Eds.), Smart Digital Futures 2014
Vol. 261. G. Phillips-Wren, S. Carlsson, A. Respício and P. Brézillon (Eds.), DSS 2.0 – Supporting Decision Making with New Technologies
Vol. 260. T. Tokuda, Y. Kiyoki, H. Jaakkola and N. Yoshida (Eds.), Information Modelling and Knowledge Bases XXV
Vol. 259. K.D. Ashley (Ed.), Legal Knowledge and Information Systems – JURIX 2013: The Twenty-Sixth Annual Conference
Vol. 258. K. Gerdes, E. Hajičová and L. Wanner (Eds.), Computational Dependency Theory
Vol. 257. M. Jaeger, T.D. Nielsen and P. Viappiani (Eds.), Twelfth Scandinavian Conference on Artificial Intelligence – SCAI 2013
Vol. 256. K. Gibert , V. Botti and R. Reig-Bolaño (Eds.), Artificial Intelligence Research and Development – Proceedings of the 16th International Conference of the Catalan Association for Artificial Intelligence
Vol. 255. R. Neves-Silva, J. Watada, G. Phillips-Wren, L.C. Jain and R.J. Howlett (Eds.), Intelligent Decision Technologies – Proceedings of the 5th KES International Conference on Intelligent Decision Technologies (KES-IDT 2013)
Vol. 254. G.A. Tsihrintzis, M. Virvou, T. Watanabe, L.C. Jain and R.J. Howlett (Eds.), Intelligent Interactive Multimedia Systems and Services

ISSN 0922-6389 (print)
ISSN 1879-8314 (online)

STAIRS 2014

Proceedings of the 7th European Starting AI Researcher Symposium

Edited by

Ulle Endriss

ILLC, University of Amsterdam

and

João Leite

CENTRIA, Universidade NOVA de Lisboa

IOS Press

Amsterdam • Berlin • Tokyo • Washington, DC

ISBN 978-1-61499-420-6 (print)
ISBN 978-1-61499-421-3 (online)
Library of Congress Control Number: 2014944111

Publisher
IOS Press BV
Nieuwe Hemweg 6B
1013 BG Amsterdam
Netherlands
fax: +31 20 687 0019
e-mail: order@iospress.nl

Distributor in the USA and Canada
IOS Press, Inc.
4502 Rachael Manor Drive
Fairfax, VA 22032
USA
fax: +1 703 323 3668
e-mail: iosbooks@iospress.com

PRINTED IN THE NETHERLANDS

STAIRS 2014
U. Endriss and J. Leite (Eds.)

Preface

These are the proceedings of the 7th European Starting AI Researcher Symposium (STAIRS), held as a satellite event of the 21st European Conference of Artificial Intelligence (ECAI) in Prague, Czech Republic, on 18th and 19th of August 2014. STAIRS is aimed at young researchers in Europe and beyond, particularly PhD students. It provides an opportunity to go through the experience of submitting to and presenting at an international event with a broad scientific scope.

The Call for Papers was soliciting submissions from all areas of AI, ranging from foundations to applications. Topics of interest for STAIRS include autonomous agents and multiagent systems; constraints, satisfiability, and search; knowledge representation, reasoning, and logic; machine learning and data mining; planning and scheduling; uncertainty in AI; natural language processing; as well as robotics, sensing, and vision. That is, the scope of STAIRS is the same as that of the major international conferences in AI. What sets STAIRS apart is that the principal author of every submitted paper must be a young researcher who either does not yet hold a PhD or who has obtained their PhD less than one year before the paper submission deadline.

We received a total of 45 submissions. All of them were carefully reviewed by the STAIRS Programme Committee, consisting of leading European researchers who together cover the depth and breadth of the field of AI. We are very grateful for the great service provided by these colleagues, as well as by the additional reviewers assisting them in their task. In the end, 16 papers were accepted for oral presentation at the symposium, and a further 14 for presentation during a poster session. These 30 accepted papers are included in this volume.

The body of submitted papers together covers the field of AI well, with knowledge representation and reasoning, machine learning, and planning and scheduling being the areas attracting the largest numbers of submissions. The problems tackled range from classical AI themes such as search, all the way to emerging research trends, e.g., at the interface of AI with economics. Also in terms of the foundations/application divide the STAIRS programme covers the full spectrum.

Besides the presentation of contributed papers and posters, the STAIRS programme will feature two keynote talks. On the first day of the symposium, Michael Wooldridge, Professor of Computer Science at the University of Oxford, UK, will offer an introduction to characterisation results for equilibria in repeated games and the significance of such results to multiagent systems. On the second day, Francesca Rossi, Professor of Computer Science at the University of Padova, Italy, will talk about new approaches to sentiment analysis, harnessing modern techniques from AI, such as preference reasoning and computational social choice. We thank both of them for accepting our invitation.

We are looking forward to an exciting two days in Prague, and we hope that readers of this volume will find it as useful as we have in getting an impression of current developments in our field.

June 2014

Ulle Endriss
João Leite

This page intentionally left blank

Symposium Organisation

Program Co-chairs

Ulle Endriss	ILLC, University of Amsterdam
João Leite	Universidade NOVA de Lisboa

Program Committee

Natasha Alechina	University of Nottingham
Jose Julio Alferes	Universidade NOVA de Lisboa
Pietro Baroni	University of Brescia
Ronen Brafman	Ben-Gurion University
Gerhard Brewka	Leipzig University
Hubie Chen	Universidad del País Vasco and Ikerbasque
Martine De Cock	Ghent University
Eric De La Clergerie	INRIA
Piotr Faliszewski	AGH University of Science and Technology
Michael Fink	Vienna University of Technology
Jörg Hoffmann	Saarland University
Paolo Liberatore	University of Rome
Weiru Liu	Queen's University Belfast
Ramon Lopez De Mantaras	IIIA – CSIC
Ines Lynce	INESC-ID/IST, University of Lisbon
Pierre Marquis	CRIL-CNRS and Universit d'Artois
Nicolas Maudet	Université Paris 6
Hector Palacios	Universidad Carlos III
David Schlangen	Bielefeld University
Stefan Schlobach	Vrije Universiteit Amsterdam
Steven Schockaert	Cardiff University
Elizabeth Sklar	University of Liverpool
Stefan Szeider	Vienna University of Technology
Ivan Titov	ILLC, University of Amsterdam
Wiebe Van Der Hoek	University of Liverpool
Stefan Woltran	Vienna University of Technology
Pinar Yolum	Bogazici University
Marius Zöllner	Karlsruhe Institute of Technology

Additional Reviewers

Kim Bauters	Frederic Moisan	Jordi Planes
Golnoosh Farnadi	Nysret Musliu	Stephanie Roussel
Ronald de Haan	Sebastian Ordyniak	Stefan Rümmele
Oliver Kutz	Andreas Pfandler	Henning Schnoor
Els Lefever		

This page intentionally left blank

Contents

This page intentionally left blank

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-1

On the Extension of Learning for Max-SAT

André ABRAMÉ and Djamal HABET
Aix Marseille Université, CNRS, ENSAM, Université de Toulon, LSIS UMR 7296, 13397, Marseille, France.
emails: {andre.abrame,djamal.habet}@lsis.org

Abstract. One of the most critical components of Branch & Bound (BnB) solvers for Max-SAT is the estimation of the lower bound. At each node of the search tree, they detect inconsistent subsets (IS) of the formula by unit propagation based methods and apply a treatment to them. The currently best performing Max-SAT BnB solvers perform a very little amount of memorization, thus the same IS may be detected and treated several times during the exploration of the search tree. We address in this paper the problem of increasing the learning performed by BnB solvers. We present new sets of clause patterns which produce unit resolvent clauses when they are transformed by max-resolution. We study experimentally the impact of these transformation' memorization in our solver AHMAXSAT and we discuss their effects on the solver behavior.

1. Introduction

The Max-SAT problem consists in finding, for a given CNF formula, a Boolean assignment of the variables of this problem which maximizes (minimizes) the number of satisfied (falsified) clauses. In the weighted version of Max-SAT, a positive weight is associated to each clause and the goal is to find an assignment which maximizes (minimizes) the sum of the weights of the satisfied (falsified) clauses. For clarity reasons, we use in this paper unweighted notations and examples. Nevertheless, all the presented results can easily be extended to weighted Max-SAT.

Among the complete methods for solving Max-SAT, Branch and Bound (BnB) algorithms (e.g. WMAXSATZ [10,8,9]) have shown their efficiency, especially on random and crafted instances. BnB solvers explore the whole search space and compare, at each node of the search tree, the current number of falsified clauses plus an (under-)estimation of the ones which will become falsified (the lower bound, LB) to the best solution found so far (the upper bound, UB). If LB $\geq$ UB, then no better solution can be found by extending the current branch and they perform a backtrack. The estimation of the remaining inconsistencies is a critical component of BnB solvers: it is one of the most time-consuming components of the solvers and its quality determines the number of explored nodes.

Efficient BnB Max-SAT solvers estimate the number of clauses which will become falsified by counting the disjoint inconsistent subsets (IS) of the formula. They use unit propagation (UP) based methods to detect inconsistencies and analyze the propagation

steps which have led to them to build inconsistent subsets. Each detected IS must be treated to ensure it will be counted only once. Two treatments are actually used by BnB solvers. If an IS matches (completely or partially) some patterns, then solvers transform it (completely or partially) by applying several max-resolution steps [1,3,4,9,5] on its clauses, and they keep the modifications in the lower nodes of the search tree. This treatment acts as a (restricted) learning or memorizing mechanism. Otherwise, they simply remove the IS's clauses from the formula or apply the max-resolution based treatment and, in both cases, they keep the modifications only at the current node of the search tree (i.e. modifications are undone before the next decision).

We propose in this paper to increase the amount of learning performed by BnB solvers. We focus on the subsets of clauses which produce, once transformed by max-resolution, unit resolvent clauses. The benefits of memorizing such transformations are double. They reduce the number of redundant propagations and max-resolution transformations. Moreover, the produced unit clauses may empower the detection of IS by unit propagation. We define new patterns corresponding to these subsets. We study experimentally the impact of their transformation's memorization on the behavior of our solver AHMAXSAT by varying the sizes of the patterns and the sizes of their clauses. The results obtained show the interest of our approach and give interesting clues on the impact of the max-resolution transformations on the solver's behavior.

2. Preliminaries

We give in this section the definitions and notations used in this paper and we present the max-resolution inference rule, which is the keystone of the learning procedure for Max-SAT.

2.1. Definitions and Notations

A formula Φ in conjunctive normal form (CNF) defined on a set of propositional variables $X = \{x_1, \ldots, x_n\}$ is a conjunction of clauses. A clause c_j is a disjunction of literals and a literal l is a variable x_i or its negation $\bar{x}_i$. Alternatively, a formula can be represented as a multiset of clauses $\Phi = \{c_1, \ldots, c_m\}$ and a clause as a set of literals $c_j = \{l_{j_1}, \ldots, l_{j_k}\}$. An assignment can be represented as a set I of literals which cannot contain both a literal and its negation. If x_i is assigned to $true$ (resp. $false$) then $x_i \in I$ (resp. $\bar{x}_i \in I$). I is a complete assignment if $|I| = n$ and it is partial otherwise. A literal l is said to be satisfied by an assignment I if $l \in I$ and falsified if $\bar{l} \in I$. A variable which does not appear either positively or negatively in I is unassigned. A clause is satisfied by I if at least one of its literals is satisfied, and it is falsified if all its literals are falsified. By convention, an empty clause (denoted by $\square$) is always falsified. Eventually, solving the Max-SAT problem consists in finding a complete assignment which maximizes (minimizes) the number of satisfied (falsified) clauses of Φ.

2.2. max-resolution rule

The max-resolution inference rule [1,3,4] is the Max-SAT version of the SAT resolution. It is defined as follows:

$$\frac{c_i = \{x, y_1, \ldots, y_s\},\ c_j = \{\bar{x}, z_1, \ldots, z_t\}}{cr = \{y_1, \ldots, y_s, z_1, \ldots, z_t\},\ cc_1,\ \ldots,\ cc_t,\ cc_{t+1},\ \ldots,\ cc_{t+s}}$$

with: $cc_1 = \{x, y_1, \ldots, y_s, \bar{z}_1, z_2, \ldots, z_t\}$, $cc_2 = \{x, y_1, \ldots, y_s, \bar{z}_2, z_3, \ldots, z_t\}$, $\ldots$, $cc_t = \{x, y_1, \ldots, y_s, \bar{z}_t\}$, $cc_{t+1} = \{\bar{x}, z_1, \ldots, z_t, \bar{y}_1, y_2, \ldots, y_s\}$, $cc_{t+2} = \{\bar{x}, z_1, \ldots, z_t, \bar{y}_2, y_3, \ldots, y_s\}$, $\ldots$, $cc_{t+s} = \{\bar{x}, z_1, \ldots, z_t, \bar{y}_s\}$. The premises of the rule are the original clauses c_i and c_j which are removed from the formula, while the conclusions are the resolvent clause cr and the compensation clauses $cc_1, \ldots, cc_{t+s}$ added to keep formula's equivalency.

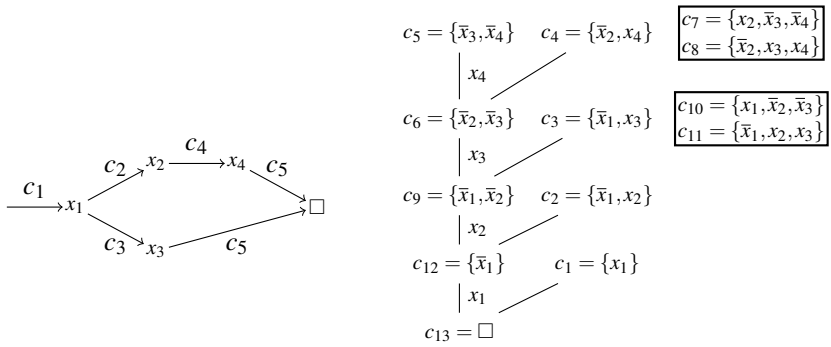
3. Learning in State of the Art BnB Solvers

At each node of the search tree, BnB solvers calculate the lower bound (LB) by estimating the number of disjoint inconsistent subsets (IS) remaining in the current formula. In this section, we first present the main techniques of IS detection and transformation. Then, we recall the learning scheme used by the best performing BnB solvers.

Recent BnB Max-SAT solvers apply unit propagation (UP) based methods to detect inconsistent subsets (more precisely simulated unit propagation (SUP) [7] and failed literals (FL) [8]). For each unit clause $\{l\}$, they remove all the occurrences of $\bar{l}$ from the clauses and all the clauses containing l. This process is repeated until an empty clause (a conflict) is found or no more unit clause remains. When an empty clause is found by UP, an inconsistent subset (IS) of the formula can be built by analyzing the propagation steps which have led to the conflict. The propagation steps made by SUP or FL can be represented by an implication graph [11], where the nodes are the assigned variables and the arrows connect the falsified literals of the unit clauses to the variables they propagate.

Once detected, an IS can be transformed by applying max-resolution operations between its clauses. The original clauses of the IS are removed from the formula, while a resolvent clause and compensation clauses are added. The resolvent is falsified by the current decisions. Thus, SUP or FL are not needed anymore to detect again the transformed IS. Consequently, by keeping the formula's transformations solvers memorize the IS and avoid its redundant detection and treatment. The following example illustrates the transformation of an IS by max-resolution.

Example 1 *Lets consider a formula* $\Phi = \{c_1, \ldots, c_5\}$ *with* $c_1 = \{x_1\}$, $c_2 = \{\bar{x}_1, x_2\}$, $c_3 = \{\bar{x}_1, x_3\}$, $c_4 = \{\bar{x}_2, x_4\}$ *and* $c_5 = \{\bar{x}_3, \bar{x}_4\}$. *The application of SUP leads to the propagation sequence* $< x_1@c_1, x_2@c_2, x_3@c_3, x_4@c_4 >$ *(meaning that* x_1 *is propagated by* c_1*, then* x_2 *by* c_2*, etc.). The clause* c_5 *is empty and the corresponding implication graph is shown on Fig. 1(a). Hence,* Φ *is an inconsistent subset. Its transformation by the max-resolution rule is done as follows. Max-resolution is first applied between* c_5 *and* c_4 *on the variable* x_4. *The intermediary resolvent* $c_6 = \{\bar{x}_2, \bar{x}_3\}$ *is produced as well as the compensation clauses* $c_7 = \{x_2, \bar{x}_3, \bar{x}_4\}$ *and* $c_8 = \{\bar{x}_2, x_3, x_4\}$. *The original clauses* c_4 *and* c_5 *are removed from the formula. Then, the max-resolution is applied between the intermediary resolvent* c_6 *and the next original clause* c_3 *on the variable* x_3 *and so forth. Fig. 1(b) shows the max-resolution steps with in boxes the compensation clauses (note that this treatment is close to the conflict analysis procedure of modern SAT solvers [11]). After complete transformation, we obtain the formula* $\Phi' = \{\square, c_7, c_8, c_{10}, c_{11}\}$ *with* $c_{10} = \{x_1, \bar{x}_2, \bar{x}_3\}$ *and* $c_{11} = \{\bar{x}_1, x_2, x_3\}$.



(a) Implication graph (b) Max-SAT resolution steps

Figure 1. Implication graph and Max-SAT resolution steps applied on the formula Φ from Example 1.

However, the Max-SAT clause learning scheme has two drawbacks which prevent its generalization. Firstly, the added resolvents and compensation clauses can increase quickly the size of the formula if learning is frequently used. Secondly, it may reduce the quality of the LB estimation and thus the number of explored nodes of the search tree may increase.

For the reasons cited above, the current best performing BnB solvers [6] keep the transformations made by the max-resolution rule only in the sub-part of the search tree (i.e. changes are undone when backtracking). Also, they restrict the application of this rule to particular sets of clauses corresponding to one (or a combination) of the following patterns (with on top the clauses of the original formula and on bottom the clauses obtained after transformation):

$$(1)\frac{\{\{x_1,x_2\},\{x_1,\bar{x}_2\}\}}{\{\{x_1\}\}},\ (2)\frac{\{\{x_1,x_2\},\{x_1,x_3\},\{\bar{x}_2,\bar{x}_3\}\}}{\{\{x_1\},\{x_1,x_2,x_3\},\{\bar{x}_1,\bar{x}_2,\bar{x}_3\}\}},\ (3)\frac{\{\{x_1\},\{\bar{x}_1,x_2\},\{\bar{x}_2,x_3\},\ldots,\{\bar{x}_{k-1},x_k\},\{\bar{x}_k\}\}}{\{\square,\{x_1,\bar{x}_2\},\{x_2,\bar{x}_3\},\ldots,\{x_{k-1},\bar{x}_k\}\}}$$

These patterns do not necessarily cover the IS entirely. Especially, the application of the max-resolution based transformation on patterns (1) or (2) produces a unit resolvent clause $\{x_1\}$. Memorizing such transformations reduce the number of redundant propagations and max-resolution steps. Moreover, the produced unit clauses can be used in the sub-part of the search tree to make further unit propagation steps, and thus it may improve the number of detected IS.

4. Extended Learning

The lower bound computation, based on the estimation of the number of disjoint inconsistent subsets, is a critical part of BnB Max-SAT solvers. Indeed, it is one of their most time-consuming components and the estimation quality guides the backtrack and thus determines the number of explored nodes in the search tree. Thus, a balance must be struck between the time spent computing this estimation and its quality. In this regard, learning seems a natural way to limit the time spent on the LB computation by making the IS detection and transformation more incremental.

We present in this section a generalization of the patterns producing unit resolvent clauses, which we name *Unit Clause Subsets*. We motivate this generalization and show that the corresponding patterns can be detected efficiently. Also, we give statistics on their occurrences in the instances of the benchmark used in Section 5.

4.1. Unit Clause Subset

We have seen in the previous section that two of the patterns used for memorization by the current best performing BnB solvers produce after transformation a unit resolvent clause. The goal of memorizing such transformations is double. On the one hand, it reduces the number of redundant propagations and on the other hand, it may empower the detection of inconsistencies in lower nodes of the search tree since more unit clauses are available for applying unit propagation.

We propose in this paper to extend the amount of learning performed by BnB solvers by considering more patterns which produce unit resolvent clauses when transformed by max-resolution. These patterns can be formally defined as follows.

Definition 1 (Unit Clause Subset (UCS)) *Let Φ be a CNF formula. A unit clause subset (UCS) is a set $\{c_{i_1},\ldots,c_{i_k}\} \subset \Phi$ with $\forall j \in \{1,\ldots,k\},\ |c_{i_j}| > 1$ such that there exists an order of application of $k-1$ max-resolution steps on $c_{i_1},\ldots,c_{i_k}$ which produces a unit resolvent clause. We denote the set of the UCS's patterns of size k by k-UCS.*

Example 2 *Below the patterns of the 3-UCS set:*

$$(2)\ \frac{\{\{x_1,x_2\},\{x_1,x_3\},\{\bar{x}_2,\bar{x}_3\}\}}{\{\{x_1\},\{x_1,x_2,x_3\},\{\bar{x}_1,\bar{x}_2,\bar{x}_3\}\}},\quad (4)\ \frac{\{\{x_1,x_2\},\{\bar{x}_2,x_3\},\{x_1,\bar{x}_2,\bar{x}_3\}\}}{\{\{x_1\},\{\bar{x}_1,\bar{x}_2,x_3\}\}},$$

$$(5)\ \frac{\{\{x_1,x_2\},\{x_1,x_3\},\{x_1,\bar{x}_2,\bar{x}_3\}\}}{\{\{x_1\},\{x_1,x_2,x_3\}\}},\quad (6)\ \frac{\{\{x_1,x_2\},\{x_1,\bar{x}_2,x_3\},\{x_1,\bar{x}_2,\bar{x}_3\}\}}{\{\{x_1\}\}}$$

Since one of the goals of memorizing transformations which produce unit resolvent clauses is to increase the number of assignments made by unit propagation (SUP or FL), we do not consider the subsets of clauses which contain unit clauses. In the best case (if they contain only one unit clause), the transformation of such subsets lets the number of unit clauses of the formula unchanged. In the worst case (if they contain more than one unit clause), the transformed formula contains less unit clauses than the original one. Thus the number of assignments made by unit propagation and consequently the number of detected IS may be reduced. Eventually, we make a distinction between the patterns of the k-UCS sets depending on the size of their clauses. We denote k^b-UCS the subset of k-UCS composed of the patterns which contain only binary clauses and k^t-UCS the subset composed of the patterns which contain at least one ternary clause. It should be noted that the patterns (1) and (2) presented in Section 3 belong respectively to the sets 2-UCS and 3^b-UCS.

4.2. Detecting k-UCS

The k-UCS patterns are easily detectable by analyzing the implication graph. Indeed, the clauses which are between the conflict and the *first unit implication point* (FUIP) [11] produce a unit resolvent clause when transformed by max-resolution. Thus, solvers

simply have to count the number and the sizes of clauses between the conflict and the FUIP to know if they are in presence of a valid UCS. This does not change the complexity of the conflict analysis procedure and the computational overhead is negligible.

4.3. Occurrences of k-UCS

We have measured the rate of occurrences of the k-UCS patterns in the inconsistent subsets detected by our solver AHMAXSAT. Tab. 1 show the results obtained on the benchmark presented in Section 5. It is interesting to observe that the patterns which are currently used for learning by state of the art solvers (2-UCS and 3^b-UCS) occur in less than 3.5% of the IS. On average, UCS are detected in 35% of the ISs. This value however varies considerably from one instance class to another.

Table 1. Percentage of occurrences of the k-UCS's patterns.

	Instances classes	2-UCS	3^b-UCS	3^t-UCS	4^b-UCS	4^t-UCS	5^b-UCS	5^t-UCS	k-UCS, $k > 5$
unweighted	crafted/bipartite	0	0.41	0	0	0.01	38.45	0.14	21.47
	crafted/maxcut	0	11.58	0	3.17	5.48	3.81	3.6	8.64
	random/highgirth	0.06	0.05	0.01	0.03	0.02	0.02	0.03	0.53
	random/max2sat	0	1.91	0	15.97	0.11	8.41	1.01	17.5
	random/max3sat	0.38	1.75	0.98	2.29	2.54	0.91	3.52	12.35
	random/min2sat	0	1.91	0	13.01	0.01	9.13	0.06	21.4
weighted	crafted/frb	0	5.36	0	0	8.05	1.17	0	5.08
	crafted/ramsey	0	1.06	0	0	0.19	0.12	0.21	0.85
	crafted/wmaxcut	0	12.99	0	0.66	8.25	3.87	5.39	10.37
	random/wmax2sat	0	2.13	0	17.69	0.09	9.56	1.21	14.58
	random/wmax3sat	0.17	1.02	0.52	1.53	1.55	0.66	2.36	8.77
	Total	0.06	3.33	0.15	6.72	1.87	8.99	1.61	13.14

5. Empirical Evaluation of UCS Learning

We present in this section an empirical evaluation of the impact of the k-UCS transformation's memorization on all the random and crafted instances from the unweighted and weighted categories of the Max-SAT Competition 2013[1]. Note that we do not include any (weighted) Partial Max-SAT instances nor industrial ones in our experiments. Even if the results presented in this paper can naturally be extended to these instance categories, our solver AHMAXSAT does not handle them efficiently. A performing BnB solver for (weighted) Partial Max-SAT must consider both the soft and the hard parts of the instances. Thus, it must include SAT mechanisms such as nogood learning, activity-based branching heuristic or backjumping and our solver currently does not [2,11]. For the industrial instances, solvers must have a very efficient memory management. None of the best performing BnB solvers (including ours) handles huge industrial instances efficiently[2].

We have implemented the UCS learning scheme in our BnB solver AHMAXSAT[3]. The experiments are performed on machines equipped with Intel Xeon 2.4 Ghz proces-

[1] Available from http://maxsat.ia.udl.cat:81/13/benchmarks.

[2] See for instance http://maxsat.ia.udl.cat:81/13/results/index.html

[3] An early version of AHMAXSAT has been submitted to the Max-SAT Competition 2013. It was the version 1.16. Since that competition, we have made numerous optimizations. The version presented in this paper is numbered 1.52.

sors and 24 Gb of RAM and running under a GNU/Linux operating system. The cutoff time is fixed to 1800 seconds per instance, as in the Max-SAT Competitions.

In the rest of this section, we present and discuss the results obtained by increasing the learning performed by AHMAXSAT. Starting from a variant using the patterns used by state of the art solvers ($\{2,3^b\}$-UCS), we add the memorization of the 3^t-UCS transformations, then the 4-UCS ones and the 5-UCS ones. Preliminary results (not reported in this paper) suggest that memorizing k-UCS transformations with $k > 5$ has a negative impact on the solver performances. The obtained results are presented in the Tables 2, 3 and 4. For each AHMAXSAT variant, we present the number of solved instances (columns S) and the averages of: numbers of decisions (columns D), solving time (columns T), number of propagations per decision (columns P/D) and number of IS detected per decision (columns □/D). The main goal of these two last indicators is to show the reduction of the redundant propagations and IS detections. They may also indicate a loss in the quality of the LB estimation, which is one of the known drawbacks of the memorization.

$\{2,3^t\}$-UCS vs. $\{2,3\}$-UCS As one can have expected, memorizing the transformations of the 3^t-UCS in addition to the ones of the patterns used by state of the art BnB solvers (i.e. $\{2,3^b\}$-UCS) does not change much the behavior of AHMAXSAT since these patterns occur very rarely in the benchmark's instances. On the instances with a non-null rate of occurrences (random/max3sat and random/wmax3sat), we can observe a slight reduction of the average number of decisions (column D) and of the average solving time (column T). On the overall benchmark, the average solving time is reduced by 2%.

$\{2,3\}$-UCS vs. $\{2,3,4\}$-UCS If we add the memorization of the transformations of the 4-UCS, both the averages of the number of propagations per decision (column P/D) and of the number of inconsistencies detected per decision (column □/D) decrease significantly (respectively -13% and -26%). Since the amount of memorization increases, one can expect a slight reduction of these two values, but not in such proportion. An important reduction probably indicates a loss in the quality of the LB estimation. This explanation is confirmed by the increase of the average number of decisions (+8% on average). Consequently, 4 less instances are solved and the solving time is on average 36% higher.

It is interesting to notice that the impact of memorizing the 4-UCS transformations vary from one instance's class to another. The loss in performances occurs on the classes with a high percentage of 4^b-UCS occurrences (random/max2sat and random/wmax2sat where the average solving time increases of 244% and 171% respectively). Conversely, on the instance's classes with a low percentage of occurrences of 4^b-UCS (crafted/frb and crafted/wmaxcut), the average solving time is reduced (-35% and -34% respectively).

$\{2,3\}$-UCS vs. $\{2,3,4^t\}$-UCS If we ignore the 4^b-UCS and memorize only the transformations of the $\{2,3,4^t\}$-UCS, both the average number of decisions and the average solving time are equal or reduced on all the instance classes (respectively -14% and -7% on average) and one more instance is solved. One can observe that the average numbers of propagations per decision and of IS detected per decision decrease slightly (respectively -0.5% and -2.5% on average).

$\{2,3,4^t\}$-UCS vs. $\{2,3,4^t,5\}$-UCS The same behavior as for the 4-UCS can be observed if we add the memorization of the transformations of the 5-UCS. Both the average number of propagations per decision and the average number of inconsistencies detected

per decision decrease significantly (-20% and -35%). Consequently, the average number of decisions increases (+42%), 7 instances less are solved and the average solving time increases of 90%. As for the 4-UCS, the loss is particularly high on instance classes with a high percentage of 5^b-UCS occurrences (crafted/bipartite, random/max2sat, random/min2sat and random/wmax2sat where the average solving time increase of respectively 235%, 322%, 188% and 261%).

Table 2. Detailed results of the variants AHMAXSAT $\{2,3^b\}$-UCS and AHMAXSAT $\{2,3\}$-UCS. The two first columns give respectively the instances classes and the number of instances per class.

Instances		#	AHMAXSAT $\{2,3^b\}$-UCS					AHMAXSAT $\{2,3\}$-UCS				
			S	D	T	P/D	□/D	S	D	T	P/D	□/D
unweighted	crafted/bipartite	100	100	35392	96.9	2267	148	100	35416	97.2	2267	148
	crafted/maxcut	67	56	235202	58.9	274	42	56	235202	59.5	274	42
	random/highgirth	82	7	4953454	1194.1	87	5	7	4991152	1199.3	87	5
	random/max2sat	100	100	40410	84.9	1860	99	100	40408	84.7	1859	99
	random/max3sat	100	98	425744	332.6	432	46	98	407917	315.6	429	45
	random/min2sat	96	96	1025	2.5	1937	64	96	1019	2.5	1937	64
weighted	crafted/frb	34	14	494542	77	79	12	14	494542	76.7	79	12
	crafted/ramsey	15	4	158350	55.4	26	10	4	158350	55.3	26	10
	crafted/wmaxcut	67	62	40645	60.6	409	169	62	40649	60.6	409	169
	random/wmax2sat	120	120	4337	54.4	4957	534	120	4307	53.7	4955	535
	random/wmax3sat	40	40	49771	138	1142	202	40	48515	134.8	1140	201
Global		821	697	157583	114.5	1902	173	697	155380	111.9	1901	173

Table 3. Detailed results of the variants AHMAXSAT $\{2,3,4\}$-UCS and AHMAXSAT $\{2,3,4^t\}$-UCS.

Instances		#	AHMAXSAT $\{2,3,4\}$-UCS					AHMAXSAT $\{2,3,4^t\}$-UCS				
			S	D	T	P/D	□/D	S	D	T	P/D	□/D
unweighted	crafted/bipartite	100	100	35456	96.9	2266	148	100	35478	97.3	2266	148
	crafted/maxcut	67	56	153539	37.5	207	30	56	153326	42.3	267	39
	random/highgirth	82	7	4960255	1186.5	88	5	6	4603981	1091.2	87	5
	random/max2sat	100	94	175425	291.8	1375	62	100	38878	81.4	1854	98
	random/max3sat	100	100	443640	319.5	401	41	100	413013	308.4	414	43
	random/min2sat	96	96	1198	2.3	1433	34	96	1025	2.5	1936	64
weighted	crafted/frb	34	14	289467	49.3	71	9	14	288458	48.6	71	9
	crafted/ramsey	15	4	158040	56.6	26	10	4	157902	55.4	26	10
	crafted/wmaxcut	67	62	21703	35.1	395	143	62	21757	35.2	397	142
	random/wmax2sat	120	120	13386	145.8	4434	353	120	4250	52.9	4957	531
	random/wmax3sat	40	40	46640	126.3	1114	192	40	45956	126.6	1126	196
Global		821	693	169317	152	1656	128	698	133829	103.7	1892	169

Table 4. Detailed results of the variants AHMAXSAT $\{2,3,4^t,5\}$-UCS and AHMAXSAT $\{2,3,4^t,5^t\}$-UCS.

Instances		#	AHMAXSAT $\{2,3,4^t,5\}$-UCS					AHMAXSAT $\{2,3,4^t,5^t\}$-UCS				
			S	D	T	P/D	□/D	S	D	T	P/D	□/D
unweighted	crafted/bipartite	100	99	185669	326,1	1365	60	100	34909	94,7	2258	147
	crafted/maxcut	67	56	188088	46,7	240	33	56	177964	43,8	251	35
	random/highgirth	82	6	4607524	1101,1	87	5	6	4597721	1102,3	87	5
	random/max2sat	100	94	214125	344,2	1327	59	100	38841	77,8	1793	91
	random/max3sat	100	100	450338	309,3	380	38	100	426143	293,5	385	39
	random/min2sat	96	96	3856	7,2	1484	37	96	979	2,4	1931	64
weighted	crafted/frb	34	14	247556	42,4	66	8	14	210245	37,2	70	8
	crafted/ramsey	15	4	157332	56,0	26	10	4	157478	55,5	26	10
	crafted/wmaxcut	67	62	21521	33,4	347	117	62	19993	30,8	346	116
	random/wmax2sat	120	120	16751	191,0	4322	333	120	3898	48,2	4844	504
	random/wmax3sat	40	40	45874	123,4	1093	187	40	44804	120,3	1100	188
Global		821	691	190109	197,2	1504	109	698	135686	99,1	1850	159

$\{2,3,4^t\}$-UCS *vs.* $\{2,3,4^t,5^t\}$-UCS As previously, if we ignore the 5^b-UCS and memorize only the transformation of the $\{2,3,4^t,5^t\}$-UCS there is no significant loss in solving time on any instance classes and the average solving time is reduced by 4,5%.

To summarize. by memorizing the transformations of the patterns of the sets $\{3^t,4^t,5^t\}$-UCS in addition to the sets $\{2,3^b\}$-UCS used by state of the art solvers, our solver AHMAXSAT solves one instance more. The average number of decisions and the average solving time are both reduced by 14%. It should be noted that the increase of the formula's size is limited and does not affect the solver efficiency.

Discussion It is commonly admitted that memorizing the transformations made by the max-resolution rule may in some cases reduce the quality of the LB estimation. However, to the best of our knowledge, the reasons of this behavior have never been properly described. The empirical study we have performed shows that the transformations of some specific patterns (i.e. the 4^b-UCS and the 5^b-UCS) seem particularly affected by this phenomenon. Moreover, the detailed statistics obtained show a correlation between a decrease of the number of propagations, a decrease of the number of detected IS and an increase of the number of decisions. Indeed, if the number of propagated variables is reduced, then less IS will be detected and the quality of the LB estimation will be reduced. Consequently, the backtracks will occur below in the search tree and more decisions will be needed to solve instances. We illustrate in the example below how the transformation of a 4^b-UCS decreases the number of propagated variables.

Example 3 *Lets consider the subset $\Phi'' = \{c_2,c_3,c_4,c_5\}$ of the formula Φ from Example 1. If we add the clause $c_{14} = \{\bar{x}_1,x_4\}$ to Φ'', there are two possible UCS: $\psi_1 = \{c_3,c_5,c_{14}\}$ and $\psi_2 = \{c_2,c_3,c_4,c_5\}$ which are respectively a 3^b-UCS and a 4^b-UCS. If ψ_1 is transformed by max-resolution, we obtain the formula $\Phi^{(3)} = \{c_2,c_4,c_{15},c_{16},c_{17}\}$ with $c_{15} = \{\bar{x}_1\}$, $c_{16} = \{\bar{x}_1,x_3,x_4\}$ and $c_{17} = \{x_1,\bar{x}_3,\bar{x}_4\}$. The assignment $x_1 = true$ leads to the following propagation sequence in $\Phi^{(3)}$: $< x_2@c_2, x_4@c_4 >$. The clause c_{15} is falsified while c_{16} and c_{17} are satisfied. If ψ_2 is transformed by max-resolution, we obtain the formula $\Phi^{(4)} = \{c_{12},c_7,c_8,c_{10},c_{11},c_{14}\}$ (this transformation is described in Example 1). The assignment $x_1 = true$ in $\Phi^{(4)}$ leads to the propagation sequence $< x_4@c_{14} >$. The clause c_{12} is falsified, c_8 and c_{10} are satisfied and c_7 and c_{11} are reduced. There is no more unit clauses, and x_2 cannot be propagated. It is interesting to notice that the two remaining reduced clauses $c_7 = \{x_2,\bar{x}_3\}$ and $c_{11} = \{x_2,x_3\}$ may lead to the propagation of x_2 if we apply the max-resolution between them. But the sole unit propagation mechanism is not sufficient to propagate x_2.*

Even if we have described here via an example how transformations may reduce the number of propagated variables and its impact on the quality of the LB estimation, the specific features of the transformed subsets of clauses concerned by this phenomenon are unclear. A thorough study of these characteristics would be of great interest to improve the BnB solvers learning procedure.

Finally, we have tested (using the protocol described previously) the two best performing BnB solvers of the Max-SAT Competition 2013: WMAXSATZ2009 and WMAXSATZ2013. The results (Tab. 5) shows that AHMAXSAT solves more instances than the two other solvers (respectively 41 and 7 more, columns S) and its average solving time is respectively 59% and 32% lower (columns T).

Table 5. Detailed comparison of AHMAXSAT $\{2,3,4^t,5^t\}$-UCS, WMAXSATZ2009 and WMAXSATZ2013.

	Instances	#	WMAXSATZ2009			WMAXSATZ2013			AHMAXSAT		
			S	D	T	S	D	T	S	D	T
unweighted	crafted/bipartite	100	99	527295	268.7	99	796983	282.3	**100**	34909	**94.7**
	crafted/maxcut	67	55	850803	97.4	55	755340	54.7	**56**	177964	**43.8**
	random/highgirth	82	0	-	-	0	-	-	**6**	4597721	1102.3
	random/max2sat	100	96	666713	288.1	100	523266	169.8	100	38841	**77.8**
	random/max3sat	100	97	2211487	381.7	100	1476192	**242.9**	100	426143	293.5
	random/min2sat	96	77	648900	185.5	96	22402	9.4	96	979	**2.4**
weighted	crafted/frb	34	9	1379041	12.2	14	1537566	62.8	14	210245	**37.2**
	crafted/ramsey	15	4	876667	93.4	4	549137	**52.6**	4	157478	55.5
	crafted/wmaxcut	67	61	75186	80.8	**63**	126254	73.5	62	19993	30.8
	random/wmax2sat	120	119	82064	288.9	120	81440	134.2	120	3898	**48.2**
	random/wmax3sat	40	40	328504	177.1	40	257175	130.7	40	44804	**120.3**
	Total	821	657	716729	240.2	691	541647	145	**698**	135686	**99.1**

6. Conclusion

We have presented in this paper new sets of patterns which produce, when transformed by max-resolution, unit resolvent clauses. The experimental study shows that the transformation' memorization of some pattern sets (namely the $\{2,3,4^t,5^t\}$-UCS) reduces significantly both the number of decisions made by our solver AHMAXSAT and its solving time. These experiments show that the transformation's memorization of the $\{4^b,5^b\}$-UCS reduces the solver's capability to detect inconsistencies and thus its performances. We have described this phenomenon, which had never been done before to the best of our knowledge. As future work, we will make a thorough study of this phenomenon to draw up a general learning framework for BnB Max-SAT solvers.

References

[1] M. L. Bonet, J. Levy, and F. Manyà, 'Resolution for max-sat', *Artificial Intelligence*, **171**(8-9), 606–618, (2007).

[2] N. Eén and N. Sorensson, 'An extensible sat-solver', in *SAT'03*, pp. 502–518. Springer, (2003).

[3] F. Heras and J. Larrosa, 'New inference rules for efficient max-sat solving', in *AAAI'06*, volume 1, pp. 68–73. AAAI Press, (2006).

[4] J. Larrosa and F. Heras, 'Resolution in max-sat and its relation to local consistency in weighted csps', in *IJCAI'05*, pp. 193–198. Morgan Kaufmann Publishers Inc., (2005).

[5] J. Larrosa, F. Heras, and S. de Givry, 'A logical approach to efficient max-sat solving', *Artificial Intelligence*, **172**(23), 204–233, (2008).

[6] C. M. Li, F. Manyà, N. Mohamedou, and J. Planes, 'Exploiting cycle structures in max-sat', in *SAT'09*, pp. 467–480, Springer Berlin / Heidelberg, (2009).

[7] C. M. Li, F. Manyà, and J. Planes, 'Exploiting unit propagation to compute lower bounds in branch and bound max-sat solvers', in *CP'05*, pp. 403–414, Springer Berlin / Heidelberg, (2005).

[8] C. M. Li, F. Manyà, and J. Planes, 'Detecting disjoint inconsistent subformulas for computing lower bounds for max-sat', in *AAAI'06*, pp. 86–91. AAAI Press, (2006).

[9] C. M. Li, F. Manyà, and J. Planes, 'New inference rules for max-sat', *Journal of Artificial Intelligence Research*, **30**, 321–359, (2007).

[10] C. M. Li, F. Many, N. Mohamedou, and J. Planes, 'Resolution-based lower bounds in maxsat', *Constraints*, **15**(4), pp. 456–484, (2010).

[11] J. P. Marques-Silva and K. A. Sakallah, 'Grasp: A search algorithm for propositional satisfiability', *IEEE Transactions on Computers*, **48**(5), pp. 506–521, (August 1999).

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-11

A Two-Levels Local Search Algorithm for Random SAT Instances with Long Clauses

André ABRAMÉ, Djamal HABET and Donia TOUMI
Aix Marseille Université, CNRS, ENSAM, Université de Toulon, LSIS UMR 7296, 13397, Marseille, France

Abstract. We present a local search algorithm exploiting two efficient solvers for SAT. The first one is based on the configuration checking strategy and the second one on an algorithm of the `Walksat` family. This new solver is dedicated to solve random k-SAT instances, such that $k \geq 4$. We have carried tests on the instances of the SAT Challenge 2012. The obtained results confirm the relevance of our approach.

Keywords. Configuration Checking, Novelty Heuristic, Random large k-SAT instances

1. Introduction

The satisfiability problem (SAT) consists of testing whether all clauses in a propositional formula F, in the Conjunctive Normal Form on a set of Boolean variables, can be satisfied by an assignment of truth values to its variables. The incomplete methods for SAT solving are generally based on the Stochastic Local Search (SLS). Starting by a randomly generated truth assignment of the variables of F, an SLS algorithm explores the search space by trying, at each step, to minimize the number of falsified clauses by flipping the truth value of a given variable. In dynamic local search algorithms [17], each clause has a dynamic weight and a variable is evaluated regarding its score. The score of a variable is the variation of the sum of the weights of the falsified clauses, if it is flipped. In the last years, the Configuration Checking (CC) strategy appears as a promising dynamic local search approach to solve SAT [4,6]. The main purpose of CC is to prevent cycling in local search by considering neighbors of variables. Two variables are neighbors if they appear in the same clause at least once. The configuration of a variable is the set of its neighbor variables and their corresponding truth values.

In this paper, we propose to improve the configuration checking strategy on random k-SAT such that $k \geq 4$. This improvement is carried out by using the `Novelty` heuristic with adaptive noise setting. `Novelty` is a powerful SLS algorithm from the `Walksat` family algorithms [11,16,19]. Our motivation is also to enhance the efficiency of diversification and intensification phases in CC-based solvers. The result of our work is a two-level SLS algorithm which we name `Ncca+`. The first stage in `Ncca+` consists of flipping a configuration changed decreasing variable (a variable with a positive score and hav-

ing the value of one of its neighbor variables changed since its last flip) [6], if it exists. Otherwise, `Ncca+` enters its second stage and applies a `Novelty+` like heuristic to select the variable to flip. Our algorithm is implemented on the basis of the powerful CC-based solver `CCASat` [4] which is the winner of the random SAT track of the SAT Challenge 2012[1]. We evaluate empirically `Ncca+` on the instances of this challenge regarding its robustness and its running time by a comparison to `CCASat`. We also compare `Ncca+` to other performing state-of-the-art SLS solvers. This empirical study confirms the efficiency of our solver. `Ncca+` also participated in the SAT competition 2013 [9] and won the bronze medal of the random SAT track[2].

The paper is organized as follows. Section 2 gives the necessary background and elements for the rest of the paper. Section 3 describes in details our contribution. Section 4 is dedicated to the experimental evaluation. Finally, we conclude in Section 5.

2. Preliminaries

2.1. Definitions and Notations

An instance F of the satisfiability problem (SAT) is defined by a pair $F = (X, C)$ such that $X = \{x_1 \cdots x_n\}$ is a set of n Boolean variables and $C = \{c_1 \cdots c_m\}$ is a set of m clauses. A clause $c_i \in C$ is a finite disjunction of literals and a literal is either a variable x_i or its negation $\neg x_i$. Two variables occurring in the same clause are neighbors. The set of the neighbors of x_i is denoted by $N(x_i)$. The size of a clause c_i is the number of its literals (denoted by $|c_i|$). If the size of each clause in C is equal to k then the instance is called $k-$SAT. An assignment is a mapping $I : X \rightarrow \{True, False\}$. I is complete if it maps all the variables of F. A clause $c_j \in C$ is satisfied by a complete assignment I iff it contains at least one satisfied literal, otherwise c_j is falsified. A model of F is an assignment that satisfies all the clauses of F. The satisfiability problem (SAT) consists of deciding if F has a model. If this is the case then F is satisfiable, otherwise F is unsatisfiable.

2.2. Stochastic Local Search for SAT

For a given CNF formula F, a basic Stochastic Local Search (SLS) algorithm for SAT starts by randomly generating a complete assignment I which may falsify some clauses. Hence, it attempts to minimize the number of falsified clauses by repeatedly repairing this assignment by flipping the value of one variable at once (changing its value from $false$ to $true$, or $true$ to $false$) until satisfying all clauses or reaching a cutoff time. In a dynamic SLS algorithm for SAT, a positive weight $w(c_j)$ is associated to each clause c_j in C and the evaluation of an assignment I is the sum of the weights of the clauses falsified under I, which we denote by $E(I)$. The score of a variable x_i is defined by $score(x_i) = E(I) - E(I_{x_i})$ where I_{x_i} is the complete assignment obtained by flipping x_i in I. If $score(x_i) > 0$ then x_i is called a decreasing variable.

SLS algorithms for SAT differ by their employed heuristic to choose the variable to flip. In this paper, we are interested in the heuristic used in `Novelty`, which is based on the `Walksat` algorithm architecture [16,19]. `Novelty`(p) selects randomly a falsified

[1] `baldur.iti.kit.edu/SAT-Challenge-2012/`
[2] `www.satcompetition.org/2013/`

clause c_j. Then it sorts the variables of c_j according to their scores breaking ties in favor of the least recently flipped variable and considers the two best variables under this sorting. If the best variable is not the most recently flipped one in c_j then `Novelty`(p) selects it for flipping. Otherwise, with probability p, it picks the second best one, and with probability $1-p$, it picks the best variable. When `Novelty`(p) gets stuck in local minima, a diversification phase is introduced in `Novelty+`(p,wp) [10] and `Novelty++`(p,dp) [12] to escape from such regions of the search space. With a probability wp, `Novelty+`(p,wp) picks randomly a variable from c_j and with probability $1-wp$ it does like `Novelty`(p). With a probability dp, `Novelty++`(p,dp) picks the least recently flipped variable in c_j and with probability $1-dp$ it acts like `Novelty`(p). In the adaptive versions of these algorithms, the values of p, wp and dp are adjusted depending on the evolution of the search. This adaptive noise setting was first introduced by Hoos [11] and has been applied in other works (see for instance [12]).

2.3. Configuration Checking for SAT

The Configuration Checking (CC) strategy for SAT considers during the search the relations between variables and their neighbors. It defines the configuration $C(x_i)$ of a variable x_i by a subset of I which is restricted to the variables of $N(x_i)$. If a variable in $C(x_i)$ has been flipped since the last flip of x_i then $C(x_i)$ is *changed*.

A typical CC-based algorithm for SAT follows the general scheme of an SLS algorithm. Its variable selection heuristic attempts first to flip those decreasing variables (with positive scores) that have their configurations changed [6]. Such variables are called *Configuration Changed Decreasing* (CCD). X_{CCD} is the set of CCD variables. Hence, a CC-based algorithm forbids the flip of a variable x_i if its configuration $C(x_i)$ has not changed since the last flip of x_i. If there is no CCD variable ($X_{CCD} = \emptyset$), an aspiration criterion is employed. The one defined by Cai and Su [6] selects a *significant decreasing variable* (SD) to be flipped. A variable x_i is SD if $score(x_i)$ is greater than some threshold g, $g > 0$. In practice, the value of g is equal to the average of clause weights in C and denoted by $\overline{w}$. We use X_{SD} to denote the set of the SD variables.

One of the most powerful algorithms based on the CC strategy is `CCASat` [7]. It selects the variable to flip as follows: if $X_{CCD} \neq \emptyset$ then it selects the CCD variable with the highest score. Else, if $X_{SD} \neq \emptyset$ then it chooses the SD variable with the highest score. In these two cases, `CCASat` is in an intensification/greedy phase. Otherwise (both X_{CCD} and X_{SD} are empty), it enters in the diversification phase which consists of choosing the least recently flipped variable appearing in a falsified clause selected randomly. We name such variables *focused random ones*.

3. The Ncca+ Algorithm for Random k-SAT Instances with $k \geq 4$

In this section, we start by giving some experimental observations on one of the most powerful CC-based solver, `CCASat` [7]. These observations have been performed on 480 k-SAT instances (120 instances for each k value in $\{4 \ldots 7\}$) of the SAT Challenge 2012. We have limited the running time of `CCASat` to 300 seconds per instance. We are interested in the nature of the flipped variables (CCD or SD). We have measured the average rates of flips of CCD variables (S_{CCD}), SD ones (S_{SD}) and focused random ones (S_{FR}).

The results are summarized in Table 1. On the whole, we can observe that 77.63% of flips are done on X_{CCD} variables and only 1.5% of flips are done on X_{SD} variables (this rate is only 0.25% for the 7-SAT instances). Finally, the focused random walk is applied for 20.87% of the performed flips and ranges from 12.06% for the 4-SAT instances to 28.47% for the 7-SAT instances.

Table 1. Observations on the flips done in `CCASat` on the k-SAT instances, $k \geq 4$, of the SAT Challenge 2012.

k	S_{CCD}	S_{SD}	S_{FR}
4	83.59	4.35	12.06
5	81.97	1.01	17.02
6	73.68	0.40	25.91
7	71.28	0.25	28.47
Average	**77.63**	**1.50**	**20.87**

According to these observations, it is clear that the aspiration criterion (which corresponds to the flip of SD variables) is rarely applied. Let us recall that this criterion selects a variable to flip with a score greater than some threshold which is the average of the clause weights, $\overline{w}$. We have measured the value of this threshold for the above instances and observed that this value remains around 1 which means that the weights of clauses do not vary significantly during the search. Hence, if there is no CCD variable, it is hard to have an SD one. These elements may explain the low values of S_{SD}.

To obtain a more accurate aspiration phase and to balance the use of the diversification phase, we propose to replace the selection of SD variables by the heuristic used in `Novelty`. Indeed, instead of selecting a SD variable when $X_{CCD} = \emptyset$, we select a variable appearing in a falsified clause selected randomly according to the `Novelty` heuristic. However, the variable scores used in `Novelty` are those obtained by considering the weights of clauses (in the original version of `Novelty`, the score of a variable x_i is equal to the number of falsified clauses which will become satisfied if x_i is flipped minus the number of satisfied clauses which will become falsified if x_i is flipped). Such adaptation of `Novelty` is close to the one used in `gNovelty+` [18].

Accordingly, we modify the heuristic used in `CCASat` to consider the integration of `Novelty`. The resulting algorithm is called `Ncca+`. It works as follows:

1. If the set of configuration changed decreasing variables is not empty ($X_{CC} \neq \emptyset$) then `Ncca+` selects a variable among X_{CC} of the highest score breaking ties in favor of the variable with the highest subscore.
2. Else, `Ncca+` updates the weights of the clauses of F according to PAWS scheme [20]. In PAWS, all clause weights are initialized to 1. Hence, with probability sp (smooth probability) and for each satisfied clause whose weight is greater than 1, PAWS decreases the weight of this clause by 1. Otherwise (with probability $1-sp$), PAWS increases by 1 the weights of all the falsified clauses.
3. With a probability dp (diversification probability), `Ncca+` selects a variable to flip according to `Novelty`(p) heuristic. Otherwise (with a probability $1-dp$), `Ncca+` selects the oldest variable in a falsified clause selected randomly.

Algorithm 1. `Ncca+` Algorithm for k-SAT instances, $k \geq 4$

Input: k-SAT formula F, $maxTries$, $maxSteps$
Output: A Satisfying assignment I, if F is SAT, or "Unknown"
for try =1 to $maxTries$ **do**
  $I \leftarrow$ randomly generated truth assignment
  Initialize clause weights to 1
  Initialize p and dp 0
  for step =1 to $maxSteps$ **do**
    if I satisfies F **then**
      return I
    end if
    $x_i \leftarrow$ `Pick_Var`(p, dp)
    Update `Novelty` probability parameters p and dp
    $I \leftarrow I$ with x_i flipped
  end for
end for
return "Unknown"

We give the general scheme of `Ncca+` in Algorithm 1. It starts by generating randomly a complete truth assignment I and while I does not satisfy the input formula F or a maximum number of flips $maxSteps$ is not reached, `Ncca+` selects a variable to flip following the heuristic detailed before.The values of the probabilities p and dp (used in `Pick_Var` function) are dynamically updated [11,13] (it is based on two parameters Φ and Θ to control the change of values of p and dp). The selected variable is flipped. All these steps (the inner loop of Algorithm 1) are repeated up to $maxTries$ (the outer loop).

4. Experimental Evaluation

This section is dedicated to the experimental evaluation of `Ncca+`. The evaluation is done on 480 random k-SAT instances (all satisfiable) from the SAT Challenge held in 2012[3]. The values of k are ranging from 4 to 7 with 120 instances per k value. Each set is also divided into 10 subsets of 12 instances with different sizes (regarding the number of the variables and the clauses). Table 2 gives the characteristics of these instances. Balint et al. [1] detail the generation and selection of these instances. We have selected these instances because they have different sizes and difficulties. The instances of the SAT Competition 2011, particularly the 5-SAT ones, are treatable rather easily by the current SLS SAT solvers. Thus, we do not include them in these experiments.

`Ncca+` is implemented in C/C++ and compiled with g++ with the compilation flags `-static -O3`. The experiments are made on a cluster of servers equipped with Intel Xeon 2.4 Ghz processors and 24 GB of RAM and running under a GNU/Linux operating system. The smooth probability sp of the PAWS scheme is $sp = 0.75$ for k-SAT with $k \in \{4,5\}$ and $sp = 0.92$ for k-SAT with $k \in \{6,7\}$ [4].

[3] `baldur.iti.kit.edu/SAT-Challenge-2012/downloads.html`

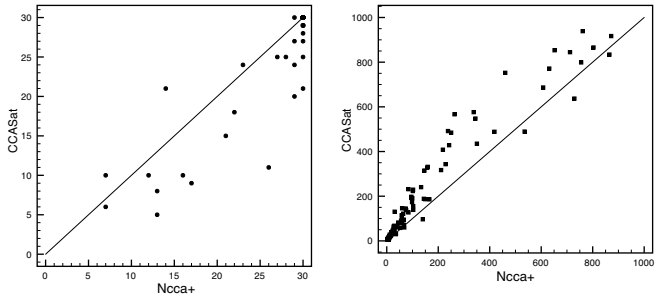
Table 2. The minimum (*min.*) and the maximum (*max.*) number of variables (n) and of clauses (m) of the k-SAT instances, $k \geq 4$, of the SAT Challenge 2012.

	4-SAT		5-SAT		6-SAT		7-SAT	
	n	m	n	m	n	m	n	m
max.	10000	90000	1600	32000	400	16000	200	17000
min.	800	7945	300	6335	200	8674	100	8779

4.1. Ncca+ vs. CCASat

Since Ncca+ is based on CCASat, we have first performed a detailed comparison of the two solvers. For this purpose, each solver is launched 30 times on each instance. Each launch terminates when a solution is found or the cutoff time of 1000 seconds is reached[4]. For each instance, we calculate the number of successful runs and the average runtime to reach a solution. If a run failed to solve an instance, a penalized time of 1000 seconds is used to compute the average time. We use two types of graphics to compare the two solvers: the first one compares the number of successful runs and consequently the solver robustness. The second one compares the average runtimes. We give and discuss these two graphs for each $k = 4 \cdots 7$ (Fig. 1 to 4) where each point in the graphs correspond to one instance. However, some plotted points may overlap if the results of the two solvers are equal or close. Finally, the parameter values of the adaptive noise mechanism are $\Phi = 10$ and $\Theta = 5$. These values are those used in competitive SLS solvers, such as TNM [13] and Sattime [14].

Random 4-SAT (Fig. 1) For these instances, we find the role of Novelty in the improvement of the robustness and the speed of CCASat. Indeed, Ncca+ enhances the robustness of the last solver on 22 instances. Also, the right graphic of Fig. 1 indicates clearly the reduction of the running time of CCASat to reach a solution. This observation is confirmed by the comparison of the average runtimes of the two solvers over the 120 4-SAT instances: 129 seconds for Ncca+ and 179 seconds for CCASat. For the instances with 3800 to 10000 variables, this time is almost divided by 2.

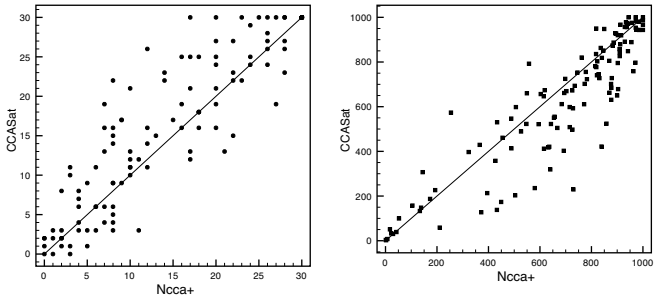


(a) Number of successful runs (b) Solving time

Figure 1. Ncca+ vs. CCASat on random 4-SAT instances.

[4]The used cutoff time during the SAT Challenge was 900 seconds under Intel Xeon 2.83 Ghz processors.

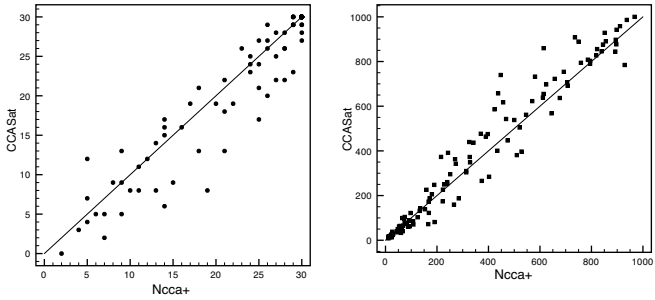
Random 5-SAT (Fig. 2) CCASat seems to be better particularly regarding the robustness criterion. It has a better success rate on 55 instances while Ncca+ is better on 40 instances. However, the average number of successful runs are very close: 15.87 successful runs for CCASat and 16.70 for Ncca+. Also, Ncca+ solved all the instances, at least once, while CCASat failed to solve 3 instances. Concerning the average runtimes over all the instances, the values are 650 and 620 seconds for Ncca+ and CCASat respectively. Hence, the last solver is 5% faster than the first one.



(a) Number of successful runs (b) Solving time

Figure 2. Ncca+ vs. CCASat on random 5-SAT instances.

Random 6-SAT (Fig. 3) Ncca+ seems to be better. Indeed, CCASat is more robust on 21 instances while Ncca+ outperforms CCASat on 33 instances. Also, over all the runs, Ncca+ successfully solved all the instances while CCASat failed to solve 1 instance. The average runtimes are 344 seconds for Ncca+ against 362 seconds for CCASat. Ncca+ solves better the instances with 200 to 300 variables, while CCASat is better on the instances with 320 and 340 variables.

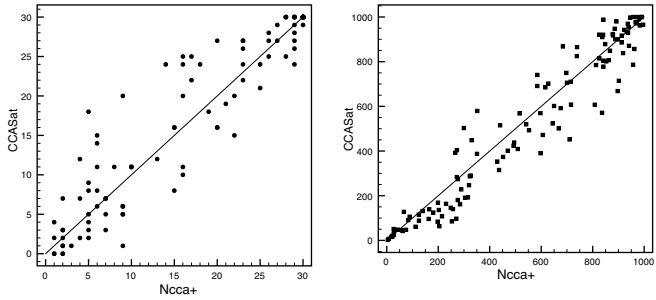


(a) Number of successful runs (b) Solving time

Figure 3. Ncca+ vs. CCASat on random 6-SAT instances.

Random 7-SAT (Fig. 4) For these instances, the results are mixed. Indeed, CCASat is more robust on 43 instances and Ncca+ on 38 instances. The average runtimes are 519 seconds for CCASat and 541 seconds for Ncca+. However, Ncca+ does better than

CCASat by solving all the instances at least one time. CCASat failed to solve 4 instances. The average number of successful runs of the two solvers over all the instances are very close: 18.57 for Ncca+ and 19.08 for CCASat.



(a) Number of successful runs (b) Solving time

Figure 4. Ncca+ vs. CCASat on random 7-SAT instances.

Over all these first results, Ncca+ seems generally better than CCASat regarding the number of solved instances. Concerning the robustness, the improvements provided by Ncca+ are clearly visible for the 4-SAT and 6-SAT instances. The same observation remains true concerning the running times. For the 5-SAT and 7-SAT instances, the results are more mixed. Nevertheless, Ncca+ solved all the instances while CCASat never reached a solution for some of them.

4.2. Ncca+ vs. State-of-the-Art SLS Solvers

In this section, we compare Ncca+ to other powerful SLS solvers including TNM [13], Sparrow2011 [2], CScoreSAT2013 [5], Sattime2013 [15], ProbSAT2013 [3] and CCASat. The solvers labelled by 2013 are taken from the SAT Competition 2013[5]. Sparrow2011 and ProbSAT2013 are the winners of the gold medal of the random SAT track of the SAT competitions 2011 and 2013 respectively. CCASat was the winner of this same category during the SAT Challenge 2012. CScoreSAT2013 is a powerful solver based on configuration checking. Sattime regularly won medals during SAT competitions of the same track. The comparison to TNM and Sattime is also motivated by the fact that these two solvers use Novelty and the set of promising decreasing variables [12] which is a subset of configuration changed decreasing variables [6].

All the solvers are run on the random instances of the SAT Challenge 2012. The cutoff time is 3600 seconds to observe the behavior of the solvers with a higher execution time than the one used during this challenge. We run each solver one time for each instance. Table 3 details the results for each k-SAT problem. It appears that Ncca+ improves CCASat for each k value, except for $k = 7$ for which CCASat solves one instance more.

[5] Available from www.satcompetition.org/2013/

Table 3. Detailed results on the 7 solvers. For each $k = 3 \cdots 7$, we give the number of solved instances by the solvers. A number in bold indicates that the solver outperforms its challengers. The numbers between brackets are the average times to find a solution for the instances in the k-SAT sets. If a solver fails to reach a solution then its runtime is penalized by 3600 seconds.

k	Ncca+	CCASat	Prob-SAT2013	CScore-SAT2013	Sattime-2013	Sparrow-2011	TNM
4	**120** **(220)**	119 (246)	119 (102)	118 (253)	61 (2250)	93 (1125)	84 (1320)
5	96 (1380)	94 (1357)	84 (1763)	**99** **(1227)**	68 (2221)	76 (1883)	41 (2819)
6	**113** (599)	108 (755)	101 (1215)	**113** **(490)**	110 (863)	87 (1688)	105 (1253)
7	103 (1090)	**104** **(1154)**	81 (1833)	96 (1192)	97 (1183)	86 (1503)	95 (1366)

For $k = 4$, Ncca+ remains better than the other solvers. For $k = 5$, CScoreSAT2013 solves 3 instances more than Ncca+ and 5 instances more than CCASat. For $k = 6$, Ncca+ and CScoreSAT2013 solve the highest number of instances (113). For $k = 7$, CCASat solves only one instance more than Ncca+ which seems to be faster. Ncca+ is clearly better than Sattime and TNM. These two last solvers also integrate the Novelty++ algorithm and they have better performances than this algorithm. Regarding all the SAT instances, Ncca+ is the better solver by solving 432 instances, the second and the third best ones are CScoreSAT2013 and CCASat which solve 426 and 425 instances respectively. We would like to note that the current scheme of Ncca+ is close to AdaptG2Wsat2009++ which alternates between greedy search thanks to the promising decreasing variables and diversification thanks to Novelty++ with adaptive noise setting [12]. We have not included the results of AdaptG2Wsat2009++ because it is outperformed by the recent solvers of the same authors, such as Sattime and TNM.

5. Conclusion

In this paper, we have presented a competitive and robust algorithm Ncca+ dedicated to random k-SAT instances with long clauses. It combines the configuration checking (CC) strategy and a heuristic similar to Novelty with adaptive noise setting. Ncca+ improved the intensification and the diversification phases used in CC-based solvers. The empirical evaluation accomplished on random instances of the SAT Challenge 2012 confirmed our purpose and the bronze medal obtained in the SAT Competition 2013 consolidated this evaluation [6].

[6] The detailed results of the SAT Competition 2013 are available from http://satcompetition.org/2013

References

[1] Balint, A., Belov, A., Järvisalo, M., Sinz, C.: Sat challenge 2012 random sat track: Description of benchmark generation. In: Proceedings of SAT Challenge 2012: Solver and Benchmark Descriptions, pp. 72–73 (2012)

[2] Balint, A., Fröhlich, A.: Improving stochastic local search for sat with a new probability distribution. In: Proceedings of SAT'10, pp. 10–15 (2010)

[3] Balint, A., Schöning, U.: Probsat. In: Proceedings of SAT Competition 2013: Solver and Benchmark Descriptions, p. 70 (2013)

[4] Cai, S., Luo, C., Su, K.: Ccasat: Solver description. In: Proceedings of SAT Challenge 2012: Solver and Benchmark Descriptions, pp. 13–14 (2012)

[5] Cai, S., Luo, C., Su, K.: Cscore2013. In: Proceedings of SAT Competition 2013: Solver and Benchmark Descriptions, pp. 18–19 (2013)

[6] Cai, S., Su, K.: Configuration checking with aspiration in local search for sat. In: Proceedings of AAAI-2012, pp. 434–440 (2012)

[7] Cai, S., Su, K.: Local search for boolean satisfiability with configuration checking and subscore. Artif. Intell. **204**, 75–98 (2013)

[8] Cai, S., Su, K., Sattar, A.: Local search with edge weighting and configuration checking heuristics for minimum vertex cover. Artif. Intell. **175**(9-10), 1672–1696 (2011)

[9] Habet, D., Toumi, D., Abramé, A.: Ncca+: Configuration checking and novelty+ like heuristic. In: Proceedings of SAT Competition 2013: Solver and Benchmark Descriptions, p. 62 (2013)

[10] Hoos, H.H.: On the run-time behaviour of stochastic local search algorithms for sat. In: Proceedings of AAAI '99/IAAI '99, pp. 661–666 (1999)

[11] Hoos, H.H.: An adaptive noise mechanism for walksat. In: Proceedings of AAAI-2002, pp. 655–660 (2002)

[12] Li, C.M., Huang, W.Q.: Diversification and determinism in local search for satisfiability. In: Proceedings of SAT'05, pp. 158–172 (2005)

[13] Li, C.M., Huang, W.Q.: Switching between two adaptive noise mechanisms in local search for sat. In: SAT 2009 competitive events booklet, p. 57 (2009)

[14] LI, C.M., LI, Y.: Satisfying versus falsifying in local search for satisfiability. In: Proceedings of SAT-2012, pp. 477–478. Springer (2012)

[15] Li, C.M., Li, Y.: Description of sattime2013. In: Proceedings of SAT Competition 2013: Solver and Benchmark Descriptions, pp. 77–78 (2013)

[16] McAllester, D., Selman, B., Kautz, H.: Evidence for invariants in local search. In: Proceedings of AAAI-1997, pp. 321–326 (1997)

[17] Morris, P.: The breakout method for escaping from local minima. In: Proceedings of AAAI'93, pp. 40–45. AAAI Press (1993)

[18] Pham, D.N., Thornton, J., Gretton, C., Sattar, A.: Combining adaptive and dynamic local search for satisfiability. JSAT **4**(2-4), 149–172 (2008)

[19] Selman, B., Kautz, H.A., Cohen, B.: Noise strategies for improving local search. In: Proceedings of AAAI-94, pp. 337–343 (1994)

[20] Thornton, J., Pham, D.N., Bain, S., Ferreira, V.: Additive versus multiplicative clause weighting for sat. In: Proceedings of AAAI'04, pp. 191–196. AAAI Press (2004)

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-21

Computing Subjective Expected Utility using Probabilistic Description Logics

Erman ACAR [a,1],
[a] *Research Group Data and Web Science, University of Mannheim, Germany*

Abstract. We introduce a framework which is based on probabilistic Description logics (Prob-DL), to represent and solve multi-criteria discrete alternative problems by calculating expected utility. To our knowledge, this is the first ever approach for calculating expected utility using a Description logics based formalism.

Keywords. Description Logics, Probabilistic Description Logic, Prob-DL, Multicriteria Decision Making, Decision Theory, Utility Theory, Subjective Expected Utility, Probabilistic Ontology

1. Introduction

Since the first serious attention of *multi-attribute utility theory* (MAUT) in [7,4] to solve problems regarding *multi-criteria decision making* (MCDM), numerous approaches have been proposed, including probabilistic, possibilistic, fuzzy and graphical models [2,15,5] amongst others. In parallel, preference representation has become an ongoing research subject in artificial intelligence, gaining more popularity every day, which also lets the discipline to deal with the problems from Decision Theory. To represent preferences and encode decision-theoretic problems, a relatively new common approach stepping forward over the last decade is the use of logical languages [16,3,18,11,12,14,13].

Description Logics (DL) is a family of logic languages which is mainly based on decidable fragments of first order logic. It has been designed to be used as a formalism in the field of knowledge representation, and it has become one of the major approaches over the last decade. In the context of the Semantic Web, it embodies a theoretical foundation for the OWL Web Ontology Language, a standard defined by the World Wide Web Consortium.

In this paper, we introduce a formal framework which is based on probabilistic Description Logic Prob-DL ([10]), a family of DL languages designed to model subjective uncertainty. The aim of our framework is to encode and solve decision problems via computing expected utility using the inference services specific to

[1]Address: Universität Mannheim, Institut für Informatik und Wirtschaftsinformatik, B6 26, Raum C1.05, D-68159 Mannheim, Germany; E-mail: erman@informatik.uni-mannheim.de

the employed language (Prob-$\mathcal{ALC}$ in our case). To our knowledge, this is also the first DL-based framework aimed to calculate the expected utility. In our approach, we represent preferences of the decision maker (agent), from the utility theory perspective, where each *criteria* has an assigned utility value (weight). We consider *alternatives* in the form of ABoxes, and criteria as concepts. We represent decision maker's background knowledge via a Prob-DL knowledge base.

The framework can be applied to *multiple criteria discrete alternative problems* (see [17]). In general, it can be applied to every domain where background knowledge which is relevant for our decisions, can be shared, matched and related via knowledge bases in terms of ontologies. One motivation is that, within a DL-based decision making framework, one can express the dependency between attributes/criteria using the concept hierarchy and evaluate an *alternative* (a *choice*) in terms of its logical implications.

In the remainder of the paper, we first briefly present preliminaries in Prob-DL, in Section 2. Then, we introduce our framework and discuss an example in Section 3. In Section 4, we discuss the related works. We conclude the paper with a brief outline and ideas about future research in Section 5.

2. Basic Prob-DL

Probabilistic Description Logics family, Prob-DL is proposed in [10] as a fragment of First-Order Logic of *Type-2* probability (see [6]). Type-2 probability refers to subjective uncertainty, or *degree of belief* e.g., "Tweety the bird flies with probability greater than 0.9", whereas Type-1 probability refers to *statistical* probability. Therefore, a probabilistic logic which solely models Type-1 probabilities, fails to represent the above statement since it can be either true or false (i.e., $Flies(Tweety)$ holds with probability of either 0 or 1).

We assume that the reader has familiarity with the basic DL [1].To introduce the basic notions and notations, following [10], we give the definition of Prob-$\mathcal{ALC}$ as a probabilistic counterpart of $\mathcal{ALC}$. N_C, N_R, N_I are denumerable sets of concept names, role names and individual names respectively. The syntax of the concepts in Prob-$\mathcal{ALC}$ extends $\mathcal{ALC}$ inductively as follows:

$$C ::= A \mid \neg C \mid C \sqcap D \mid \exists r.C \mid P_{\geq n}C \mid \exists P_{rel\ n} r.C \tag{1}$$

where $A \in N_C$, C and D are concepts, $r \in N_R$, $rel \in \{\geq, >\}$ and $n \in [0, 1]$. $C \sqcup D$ is an abbreviation for $\neg(\neg C \sqcap \neg D)$, $\forall r.C$ for $\neg\exists.\neg C$, $\top$ for $C \sqcup \neg C$ and $\bot$ for $\neg\top$. Furthermore, $P_{<n}C$ is an abbreviation for $\neg P_{\geq n}C$, $P_{\leq n}C$ for $P_{\geq 1-n}\neg C$, and $P_{>n}C$ is for $P_{<1-n}\neg C$. A TBox is a finite set of axioms (concept inclusions) $C \sqsubseteq D$, which represents the ontology. A probabilistic ABox $\mathcal{A}$ is defined according to the following rule

$$\mathcal{A} ::= C(a) \mid r(a, b) \mid \neg\mathcal{A} \mid \mathcal{A} \wedge \mathcal{A}' \mid P_{\geq n}\mathcal{A} \tag{2}$$

where $C \in N_C$, $r \in N_R$, $a, b \in N_I$, $n \in [0, 1]$, $\mathcal{A}$ and $\mathcal{A}'$ ranges over probabilistic ABoxes. Abbreviations (i.e., $P_{rel\ n}A$) are defined similarly as for concepts. A *knowledge base* $\mathcal{K}$ is a pair $(\mathcal{T}, \mathcal{A})$ where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox.

The semantics of Prob-DL is defined by generalizing the standard semantics of DL . In particular, a *probabilistic interpretation* has the form

$$\mathcal{I} = (\Delta^{\mathcal{I}}, W, (I_w)_{w \in W}, \mu), \tag{3}$$

where $\Delta^{\mathcal{I}}$ is the non-empty domain, W is a non-empty set of *possible worlds*, μ is a discrete probability distribution on W, and for each $w \in W$, $\mathcal{I}_w$ is a classical DL interpretation with domain $\Delta^{\mathcal{I}}$. It is supposed that $a^{\mathcal{I}_w} = a^{\mathcal{I}_{w'}}$ for all $a \in N_I$ and $w, w' \in W$, therefore we write $a^{\mathcal{I}}$ in short. For $A \in N_C$, the probability that $a \in \Delta^{\mathcal{I}}$ is an A, is defined as

$$p^{\mathcal{I}}_a(A) = \mu(\{w \in W \mid a \in A^{\mathcal{I}_w}\}). \tag{4}$$

Similarly, for $r \in N_R$, the probability that $a, b \in \Delta^{\mathcal{I}}$ are related by r, is defined as

$$p^{\mathcal{I}}_{a,b}(r) = \mu(\{w \in W \mid (a, b) \in r^{\mathcal{I}_w}\}). \tag{5}$$

This is extended to complex concepts C, by defining the extension $C^{\mathcal{I}_w}$ of complex concepts by mutual recursion on C. The definition of $p^{\mathcal{I}}_a(C)$ is exactly as above (i.e., A is replaced by C), and as the case for non-probabilistic concepts are defined in parallel to classical-DLs (e.g., $(C \sqcap D)^{\mathcal{I}_w} = C^{\mathcal{I}_w} \sqcap D^{\mathcal{I}_w}$), we give only the cases with probabilistic concepts:

$$\begin{aligned} (P_{rel\ n} C)^{\mathcal{I}_w} &= \{a \in \Delta^{\mathcal{I}} \mid p^{\mathcal{I}}_a(C) rel\ n\} \\ (\exists P_{rel\ n}.C)^{\mathcal{I}_w} &= \{\exists b \in C^{\mathcal{I}_w} : p^{\mathcal{I}}_{a,b}(r) rel\ n\} \end{aligned} \tag{6}$$

A probabilistic interpretation $\mathcal{I}$ satisfies a concept inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}_w} \subseteq D^{\mathcal{I}_w}$ for all $w \in W$. The interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ if it satisfies all inclusions in $\mathcal{T}$. Similarly, $\mathcal{I}_w$ satisfies assertions parallel to classical DLs (i.e., $\mathcal{I}_w \models C(a)$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}_w}$), and this is defined inductively for ABoxes. Again we provide the probabilistic case;

$$\mathcal{I}_w \models P_{rel\ n}(\mathcal{A}) \text{ iff } p^{\mathcal{I}}(\mathcal{A}) rel\ n \tag{7}$$

where $p^{\mathcal{I}}(\mathcal{A})$ is the probability that an ABox A holds and it is defined as

$$p^{\mathcal{I}}(\mathcal{A}) = \mu(\{w \in W \mid \mathcal{I}_w \models \mathcal{A}\}). \tag{8}$$

Note that, in this semantics $P_{rel\ n}(C(a))$ and $(P_{rel\ n} C)(a)$ are equivalent. It is said that $\mathcal{I}$ is a model of $\mathcal{A}$ if $\mathcal{I}_w \models \mathcal{A}$ for some w, and is a model of $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if it is a model of both $\mathcal{T}$ and $\mathcal{A}$. A knowledge base $\mathcal{K}$ is *consistent* if it has a model. For convenience, we restrict ourselves to the language Prob-$\mathcal{ALC}_c$ which does not allow probabilistic roles, that is $\exists P_{rel\ n}$.C. This is because Prob-$\mathcal{ALC}_c$ is expressive enough for our purpose and also expectedly it provides much better complexity results (the consistency in full Prob-$\mathcal{ALC}$ is 2-EXPTIME-hard, whereas

for Prob-$\mathcal{ALC}_c$ it is EXPTIME-complete) [10]. For the procedure for consistency check and details we refer the reader to [10].

As mentioned in [10], standard semantics of Prob-$\mathcal{ALC}$ does not support deducing the probability of *independent events* (i.e., $p(A \wedge B) = p(A) \cdot p(B)$). For that reason, Prob-$\mathcal{ALC}^{\text{indep}}$ is introduced (see [10]) as an extension of Prob-$\mathcal{ALC}$, which allows *independence constraints* in the form of indep(C, D) in the TBox; C, D being concepts, $p_d^{\mathcal{I}}(C) \cdot p_d^{\mathcal{I}}(D) = p_d^{\mathcal{I}}(C \sqcap D)$.

3. Decision Bases and Expected Utility

We model a discrete multi-criteria decision problem from the agent's perspective in the sense that *in the light of background knowledge and ranked outcomes which choice (alternative) should the agent take?* Here, ranking of outcomes are projected by agent's subjective utility values, and background knowledge is represented by a probabilistic knowledge base. We note that, although we assume the basic Prob-$\mathcal{ALC}_c$ as our base for clarity, the main working principle of our framework is not based on a specific Prob-DL language. We also note that, in this paper we do not concern ourselves with problems regarding elicitation. We assume that agent's preferences are sufficiently elicited.

3.1. Representing Discrete Multicriteria Decision Problems

We represent the background knowledge of the agent by the Prob-DL knowledge base $\mathcal{K}$, which includes the hierarchy $\mathcal{T}$ of probabilistic concepts and *assertions* about individuals which are represented in $\mathcal{A}$. The choice set $\mathcal{C}$ represents *a priori* alternatives which utilities yet are unknown to the agent. $\mathcal{U}$ is the criteria set where each of its element consists of non-probabilistic concept denoted by Y_i and a corresponding value denoted by u_i.

Definition 1 (Decision Base)**.** *A decision base, $\mathcal{D} = (\mathcal{K}, \mathcal{C}, \mathcal{U})$ is a triple with;*

- $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ *is Prob-DL* knowledge base *(background knowledge) in which $\mathcal{T}$ is a probabilistic general acyclic TBox and $\mathcal{A}$ is a probabilistic ABox,*
- $\mathcal{C} = \{Ch_1, \ldots, Ch_n\}$ *is a* choice box*, a non-empty finite set of* choices*, each being a probabilistic ABox,*
- $\mathcal{U} = \{\langle Y_1, u_1\rangle, \ldots, \langle Y_m, u_m\rangle\}$ *is a* utility box *(UBox), a finite set of non-probabilistic concepts Y_i (criterion) and a corresponding* basic utility $u_i \in \mathbb{R}^+$ *with $Y_i \equiv_{\mathcal{T}} Y_j \implies u_i = u_j$,*

with the restrictions that no nested probabilistic constructor occurs in $\mathcal{D}$ (e.g., $P_{\geq n}(P_{>n}(C))$), and for each probabilistic concept $P_{rel\ n}C$, $rel \in \{\geq, >\}$ and $n \in (0, 1]$.

In a description logic decision base, we refer to a choice or alternative, as a list of specifications about individual(s) (DL), e.g., for a car buyer, choices can be technical specifications about cars whereas for a medical doctor, they can be treatment/medication alternatives. Notice that, one could also give an alternative definition that allows probabilistic concepts to occur in UBox. However, this in

turn yields so much expressivity which is not very intuitive and without immediate obvious benefits. Also, we did not prefer to allow concept or role assertions in UBox, for a simpler and intuitive exposition of the framework (including the sequel). However, to make distinction between individuals and provide more expressivity from the utility perspective, one can extend the definition. Recall that basic utility values are solely subjective, serving our purpose. Also, we have restricted them to be non-negative reals for the sake of a simpler exposition in the sequel. This restriction can optionally be removed to model a particular decision problem conveniently.

3.2. Subjective Expected Utility

Given that Prob-DL is developed to represent subjective uncertainty, and basic utility values for each criterion is specified, now we can define the subjective expected utility of a choice. For that purpose, let us introduce few useful notions:

- $clash_{\mathcal{K}}(C) = \{C'(a) \mid \{C(a), C'(a)\}$ is inconsistent : $\mathcal{K} \models C'(a) \wedge a \in Ind(\mathcal{A})\}$ where $Ind(\mathcal{A})$ is the set of individuals occuring in $\mathcal{A}$,
- The function $int : \mathcal{A} \rightarrow 2^{[0,1]}$ for $int(C(a)) = [n, 1]$ if C is in the form of $P_{\geq\ n}D(a)$; $int(C(a)) = [1, 1]$ if C is a non-probabilistic concept. Similarly, for the concepts with other probabilistic concept constructors (e.g., $int(C(a)) = (n, 1]$ if C is in the form of $P_{>\ n}D(a)$),
- $\wp^{rel}(\mathcal{A}) = \inf\{\bigcap_{C(a)\in\mathcal{A}}\{int(C(a))\}\}$ where $rel \in \{\geq, >\}$,

Informally, for any given concept C, $clash_{\mathcal{K}}(C)$ denotes the set of all entailed assertions (from knowledge base $\mathcal{K}$) which yields a clash. A clash is considered as the form of $[C(a), \neg C(a))]$, or in particular $[P_{<\alpha}C(a), P_{\geq\beta}C(a)]$ where $\beta \geq \alpha$, due to abbreviations of probability constructors. The function int outputs the probability interval for a given assertion. The function $\wp^{rel}$ gives the infimum of the intersection interval of all concept assertions in a given ABox. Contrary to *Analysis*, we leave infimum of an empty set as undefined.

Definition 2 (Expected Utility)**.** *The* expected utility U_{rel} *of a choice* Ch *w.r.t* $\mathcal{D} = (\mathcal{K}, \mathcal{C}, \mathcal{U})$ *is,*

$$U_{rel}(Ch) = \Sigma_{\{\langle Y_i, u\rangle \in \mathcal{U}\}} \wp^{rel}(clash_{\mathcal{K}\cup Ch}(P_{\leq 0}Y)) \cdot u$$

where $Ch \in \mathcal{C}$, $\mathcal{K} \cup Ch$ *is consistent and* $K \cup Ch \cup P_{\leq 0}Y(a)$ *is inconsistent for an* $a \in Ind(Ch \cup \mathcal{A})$, $rel \in \{\geq, >\}$.

Informally, (subjective) expected utility of a choice is the sum of the products between the infimum of the intersection of intervals of probabilities that criteria are satisfied, and basic utilities of criteria in $\mathcal{U}$, that are satisfied by that choice w.r.t. the knowledge base. Intuitively, a knowledge base and a choice ($\mathcal{K} \cup Ch$) will entail a criterion Y with a degree of probability more than zero, if $P_{\leq 0}Y$ yields inconsistency (w.r.t. $\mathcal{K} \cup Ch$) for any $a \in Ind(Ch \cup \mathcal{A})$. Semantically, $U_{rel}(Ch)$ is interpreted as rel utility of a choice, e.g., $U_{\geq}(Ch) = 40$ means that Ch has utility of at least or equal to 40. Note that U_{rel} yields a complete and transitive *preference relation* $\succeq$ over choices;

$$Ch_1 \succeq Ch_2 \iff U_{rel}(Ch_1) \geq U_{rel}(Ch_2). \tag{9}$$

In a similar fashion, an upper bound for the expected utility w.r.t. $<, \leq$ can be defined via the help of using sup in $\wp^{rel}$ instead. We conjecture that the use of *sup* and inf can induce a characterisation of *risk-seeking* and *risk averse* agents respectively, once the *maximum expected utility* is defined. We leave this to future work. In the sequel, we will drop rel, and write just U instead for the sake of simplicity.

From the definition above, it follows that the utility of an inconsistent alternative/choice (with respect to the knowledge base) is undefined. Thus we restrict ourselves to assess only the consistent decisions. This naturally provides us a service to eliminate alternatives which can cause inconsistencies.

Notice that calculating the utility of a choice, can be thought of as answering a series of consistency checking problems. We speculate that it is at least of the complexity of the consistency checking problem (of the employed Prob-DL language) in the size of the given UBox. We leave a detailed formal investigation on complexity issues to future work.

Now, given the decision base and the expected utility of a choice Ch, one can easily define the type of the problem such as calculating the **maximum expected utility**:

$$Ch_{max} = \arg\max_{Ch}\{U(Ch) \mid Ch \in \mathcal{C}\} \tag{10}$$

This can be generalised in terms of picking up the best n choices together.

$$Ch^n_{max} = \arg\max_{(Ch_1,\ldots,Ch_n)}\{U(\bigcup_{i=1}^{n} Ch_i) \mid Ch_1,\ldots,Ch_n \in \mathcal{C} \text{ and } n \leq |\mathcal{C}|\} \tag{11}$$

Or it can be logically restricted to a situation that agent can pick up at most one choice (mutually exclusive), with the following definition.

Definition 3 (Mutual Exclusion)**.** *A decision base $\mathcal{D} = (\mathcal{K}, \mathcal{C}, \mathcal{U})$ is* mutually exclusive *if for every $Ch_i, Ch_j \in \mathcal{C}$ with $i \neq j$, $Ch_i \cup Ch_j \cup \mathcal{K}$ is inconsistent.*

In general, in order to model the concerned type of a decision problem, one can bring some restrictions on $\mathcal{C}$ and $\mathcal{U}$. Also, in $\mathcal{U}$ one can express *complement attributes* that is the utility of having both attribute is greater than sum of each, e.g, $\langle TechnicallySkilled, 20\rangle$ and $\langle FluentInEnglish, 30\rangle$ whereas $\langle TechnicallySkilled \sqsubseteq FluentInEnglish, 70\rangle$. Similarly one can express *substitute attributes* i.e., having both attribute has a lower basic utility than each.

3.3. Example: Hiring an employee

We give an example on an agent (employer) giving a decision on hiring an employee based on criteria such as friendliness, punctuality, being technically skilled

$$
\begin{aligned}
\mathcal{T} = \quad & \{\exists hasPassed.Toefl \sqsubseteq P_{\geq 0.7} FluentInEnglish, \\
& Australian \sqsubseteq FluentInEnglish, \\
& \exists hasStudied.Math \sqsubseteq P_{\geq 0.9} TechnicallySkilled, \\
& \exists hasStudied.Business \sqsubseteq P_{\geq 0.4} TechnicallySkilled, \\
& \exists StudiedIn.EliteUni \sqsubseteq P_{\geq 0.8} Smart, \\
& \exists StudiedIn.AverageUni \sqsubseteq P_{\geq 0.5} Smart, \\
& P_{\geq 0.8} Friendly \sqcap Punctual \sqsubseteq Reliable\} \\
\mathcal{A} = \quad & \{EliteUni(UniB), \\
& AverageUni(UniA)
\end{aligned}
$$

$$
\begin{aligned}
Ch_1 = \{ \quad & StudiedIn(Alice, UniB), \\
& Australian(Alice), \\
& hasStudied.Business(Alice), \\
& Punctual(Alice), \\
& P_{\geq 0.8} Friendly(Alice)\}
\end{aligned}
\qquad
\begin{aligned}
Ch_2 = \{ \quad & StudiedIn(Bob, UniA), \\
& hasStudied.Math(Bob), \\
& hasPassedToefl(Bob)\}
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{U} = \{ \quad & \langle TechnicallySkilled, 90\rangle, \\
& \langle FluentInEnglish, 70\rangle \\
& \langle Smart, 55\rangle, \\
& \langle Friendly, 20\rangle, \\
& \langle Reliable, 60\rangle\}
\end{aligned}
$$

Figure 1. Employer's background knowledge $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, choices set CBox $\mathcal{C} = \{Ch_1, Ch_2\}$ representing two candidates for the *job position*, and UBox $\mathcal{U}$ on criteria and respective weights representing the preferences for the *position*.

etc. It can be thought of as impressions of the employer about two candidates after interviews.

Some of the information (subjective) given in Figure 1, is if someone studied in a elite university, she is smart at least with the probability 0.8. If an *individual* is known to be an Australian than she is certainly fluent in English. Moreover it is defined that $P_{\geq 0.8}Friendly(Alice)$ and a *Punctual individual* is *Reliable*. It can be interpreted that the agent also has the impression that Alice is punctual (which might follow from a possible scenario that she appeared on time to the interview) etc.

The interested reader can check that the expected utility of both choices and see that $Ch_1 \succeq Ch_2$ since $U(Ch_1) \geq 70 + 90 \times 0.4 + 55 \times 0.8 + 0.8 \times 20 + 60 = 226$ whereas $U(Ch_2) \geq 0.9 \times 90 + 0.5 \times 55 + 0.7 \times 70 = 157$, which is not a surprise as Bob fails to satisfy reliability and friendliness.

Notice that using Prob-$\mathcal{ALC}_c^{\text{indep}}$, one can calculate the expected utility w.r.t. an extended UBox including a criterion such as $\langle TechnicallySkilled \sqcap FluentInEnglish, 100\rangle$. In this case, the probability of $TechnicallySkilled \sqcap FluentInEnglish$ is implicitly inferred to be ≥ 0.63 for Bob, in turn, getting an additional score of 63.

4. Related Work

Preference representation using logical languages has become popular over the last decade. Many of these approaches are based on propositional logic [3,8,18]. DL languages are used for preference representation in [9,11,12,14,13,16], yet most of them are not including uncertainty. Regarding the first use of DL in the context of MAUT, [14,13], Ragone et al focus on multi-issue bilateral negotiation. In [11, 12], they mainly discuss how to compute utilities (without uncertainty), where preferences are represented by weighted DL-formulas (*preference set*), just as UBox in our approach.

According to their terminology, our approach can be understood as an *implication-based* approach. They define logical implication in terms of membership, i.e., $m \models C$ iff $m \in C^{\mathcal{I}}$. The *minimal model* that they introduced in order to define the *minimal utility value* is more restrictive than ordinary models in DL. They change this definition to ordinary models in their next paper [12], while keeping the formal machinery the same (except the way they compute utilities). Hence, in addition to not dealing with uncertainty, the main difference of our approach is the formal extension to multiple alternatives and the use of ABoxes, which provides considerable expressivity.

To our knowledge, the only work which attempts to ground MCDM problems to (fuzzy) DL formally is [16]. The main difference is the choice of fuzzy DL as formalism to deal with uncertainty. Although the terms *utility* and *preference* are not explicitly used, it consists of preferences implicitly. They base their work on a standard MCDM feature, a *decision matrix* wherein the performance score of each alternative over each criteria is explicitly stated. Criteria are expressed as fuzzy concepts. Among alternatives, the optimal alternative (w.r.t the fuzzy knowledge base) is the one with the highest *maximum satisfiability degree*. In the

explained framework, authors do not explicitly make a distinction between the knowledge base and the set of criteria. In general, the focus of the work is to show the potential and flexibility of fuzzy DL in encompassing the usual numerical methods used in MCDM, rather than leveraging the practicality of description logics in MCDM for expressing relations and handling inconsistencies between criteria, alternatives, and the knowledge base.

5. Conclusion and Further Plan

We have introduced a description logic based framework, to effectively express and solve decision problems of multi-attribute discrete alternatives.

As the major part of the utility theory and decision making literature is concerned with uncertainty, we based our approach on probabilistic description logics Prob-DL ([10]). In particular, the probabilistic extension allowed us to compute the subjective expected utility of choices in terms of their logical implications. This will allow us in our further work, to access the essential utility theory literature from the DL perspective, along with lots of new application possibilities.

One major direction is to investigate the decision theoretic properties of the utility function U, and the expressivity of $\mathcal{D}$, defining some restrictions on the UBox. Another major research direction is to extend the framework to sequential decisions (e.g. $\mathcal{D}_i \rightarrow \mathcal{D}_{i+1}$, sequence of decision bases). Once sequential decisions are defined, we will be able to represent policies and define a planner. Furthermore, it can be extended to represent collaborative decision making scenarios as well as game theoretical set-ups by considering more than one agent and specifying restrictions between their choice sets and knowledge bases. For instance in an arbitrary set-up, rules of the game could be a subset of intersection of both agent's knowledge bases, then the knowledge bases would get extended according to each players choices if each player can see what others choose. It can be checked whether a game-theoretical condition is satisfied, in terms of some corresponding conditions on ontologies.

Currently, we are working on the implementation of the basic framework as a Protégé[2] plug-in. Our plugin is planned to consist of an editor for the definition of UBoxes and choices, while the background knowledge is loaded via the standard interfaces of Protégé. Our extension will then be able to compute the utility of the given choices w.r.t background knowledge and display a ranking of choices. The development of our Protégé plugin is motivated by the idea to demonstrate the benefits of our approach to a set of different application scenarios where decision making is involved.

References

[1] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[2] `http://protege.stanford.edu/`

[2] Urszula Chajewska, Daphne Koller, and Ronald Parr. Making rational decisions using adaptive utility elicitation. In *Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00)*, pages 363–369, Menlo Park, CA, July 30– 3 2000. AAAI Press.

[3] Yann Chevaleyre, Ulle Endriss, and Jérôme Lang. Expressive power of weighted propositional formulas for cardinal preference modelling, December 08 2006.

[4] P. C. Fishburn. Interdependence and additivity in multivariate, unidimensional expected utility theory. *Intl. Economic Review*, 8:335342, 1967.

[5] Phan H. Giang and Prakash P. Shenoy. Two axiomatic approaches to decision making using possibility theory. *European Journal of Operational Research*, 162(2):450–467, April 16 2005.

[6] J. Y. Halpern. An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350, 1990.

[7] R.L. Keeney and H. Raiffa. *Decisions with multiple objectives: Preferences and value tradeoffs*. J. Wiley, New York, 1976.

[8] Céline Lafage and Jérôme Lang. Logical representation of preferences for group decision making. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *KR2000: Principles of Knowledge Representation and Reasoning*, pages 457–468, San Francisco, 2000. Morgan Kaufmann.

[9] Thomas Lukasiewicz and Jörg Schellhase. Variable-strength conditional preferences for ranking objects in ontologies. *J. Web Sem*, 5(3):180–194, 2007.

[10] Carsten Lutz and Lutz Schröder. Probabilistic description logics for subjective uncertainty. In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9-13, 2010*. AAAI Press, 2010.

[11] Azzurra Ragone, Tommaso Di Noia, Francesco M. Donini, Eugenio Di Sciascio, and Michael P. Wellman. Computing utility from weighted description logic preference formulas. In Matteo Baldoni, Jamal Bentahar, M. Birna van Riemsdijk, and John Lloyd, editors, *DALT*, volume 5948 of *Lecture Notes in Computer Science*, pages 158–173. Springer, 2009.

[12] Azzurra Ragone, Tommaso Di Noia, Francesco M. Donini, Eugenio Di Sciascio, and Michael P. Wellman. Weighted description logics preference formulas for multiattribute negotiation. In Lluis Godo and Andrea Pugliese, editors, *Scalable Uncertainty Management, Third International Conference, SUM 2009, Washington, DC, USA, September 28-30, 2009. Proceedings*, volume 5785 of *Lecture Notes in Computer Science*, pages 193–205. Springer, 2009.

[13] Azzurra Ragone, Tommaso Di Noia, Eugenio Di Sciascio, and Francesco M. Donini. Description logics for multi-issue bilateral negotiation with incomplete information. In *AAAI*, pages 477–482. AAAI Press, 2007.

[14] Azzurra Ragone, Tommaso Di Noia, Eugenio Di Sciascio, and Francesco M. Donini. DL-based alternating-offers protocol for automated multi-issue bilateral negotiation. In Diego Calvanese, Enrico Franconi, Volker Haarslev, Domenico Lembo, Boris Motik, Anni-Yasmin Turhan, and Sergio Tessaris, editors, *Proceedings of the 2007 International Workshop on Description Logics (DL2007), Brixen-Bressanone, near Bozen-Bolzano, Italy, 8-10 June, 2007*, volume 250 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[15] Yoav Shoham. Conditional utility, utility independence, and utility networks. *CoRR*, abs/1302.1568, 2013.

[16] Umberto Straccia. Multi criteria decision making in fuzzy description logics: A first step. In Juan D. Velásquez, Sebastián A. Ríos, Robert J. Howlett, and Lakhmi C. Jain, editors, *KES (1)*, volume 5711 of *Lecture Notes in Computer Science*, pages 78–86. Springer, 2009.

[17] Jyrki Wallenius, James S. Dyer, Peter C. Fishburn, Ralph E. Steuer, Stanley Zionts, and Kalyanmoy Deb. Multiple criteria decision making, multiattribute utility theory: Recent accomplishments and what lies ahead. *Management Science*, 54(7):1336–1349, 2008.

[18] Dongmo Zhang and Yan Zhang. A computational model of logic-based negotiation. In *AAAI*, pages 728–733. AAAI Press, 2006.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-31

Towards modeling surprise in economics and finance: a cognitive science perspective

Davi Baccan [a,1] , Luis Macedo [a] and Elton Sbruzzi [b]
[a] *CISUC, Department of Informatics Engineering, University of Coimbra, Coimbra, Portugal*
[b] *CCFEA, University of Essex, Essex, UK*

Abstract. In financial markets, market participants need to cope with risk and uncertainty, to forecast possible scenarios, and to constantly analyze and revise their beliefs, expectations, and strategies in the light of the massive amount of economical and financial information they receive. Interestingly, the relevance does not seem to reside in the numbers itself but rather whether they elicit "surprise" for market participants. In this paper we review the presence of the term surprise in economics and finance as well as how it is computed. Then, we present how emotions are defined in cognitive science, provide a formal definition of surprise, and describe the surprise process. Additionally, we present some theories regarding how artificial surprise can be computed. In a case study we compare the two different perspectives on surprise, discussing some similarities and differences. Finally, we present some possible applications of the cognitive science perspective.

Keywords. Autonomous agents and multiagent systems, cognitive modeling, multi-agent-based simulation, multidisciplinary topics, social simulation and modeling

1. Introduction

One of the essential tasks in the context of economics and finance is forecasting. Perhaps one of the best contexts to observe the importance of forecasting as well as the effects of a good or bad forecast is the context presented by financial markets, especially stock markets. In a stock market, market participants need to cope with risk and uncertainty [16], by assessing and, ultimately, attributing probabilities to the occurrence of a potentially good or bad/risky/unexpected event. This kind of assessment is essential because a series of strategies depend somewhat on the probability of the occurrence of events that may have an impact on asset prices [22]. Additionally, participants tend to typically trade based on the risk-return trade-off, i.e., lower (higher) levels of risk are generally associated with lower (higher) levels of potential returns.

In this complex and dynamic environment, market participants need to constantly analyze and revise their beliefs, expectations, and strategies in the light of the massive

[1]Davi Baccan is supported by TribeCA ("Trading and investing with behavioral-emotional Cognitive Agents") project, funded by the FEDER through the "Programa Operacional Regional do Centro".

amount of information they receive [29]. Such information includes a wide variety of financial and economic indicators, and data regarding companies. Interestingly, what seems to be relevant for asset behaviour and prices are not the numbers in itself, but actually whether they are lower, equal, or higher than the general beliefs and expectations, the so-called market consensus (e.g., see [2]). Terms such as "lower (higher) than expected" and "beat (miss) the expectation" are indeed commonly employed in the context of stock markets.

In the literature related to economics and finance there are several different theories and hypotheses that try to explain how a stock market works (e.g., [17, 9]). Traditional economic theories, such as the Efficient Market Hypothesis (EMH) (e.g., [9]), relies on the assumption that market participants have stable and well-defined preferences [28]. Additionally, such theories assume that when participants are confronted with decisions that involve risk, they are able to correctly form their probabilistic assessments according to the laws of probability, calculating which of the alternative courses of action maximize their expected utility. Therefore, most of the participants are efficient in reflecting new information accurately [21]. The EMH refers to hypothesis that market prices fully and instantaneously incorporate the information and expectations of all market participants. Last but not least, the EMH assumes that market participants have no cost in acquiring and analyzing information.

However, behavioral economics (e.g., [15]), i.e., the combination of psychology and economics that aims to understand human decision-making under risk as well as how this behaviour matters in economic contexts, have documented extensive experimental evidence that there are deviations from the rational behaviour. Such deviations, known as behavioral biases, are believed to be ubiquitous to humans, and several of them are clearly counterproductive from the economics perspective. Examples of behavioral biases are herding, and overreaction [5] (i.e., the evidence that most people tend to "overreact" to unexpected and dramatic news events).

In this paper we review in Section 2 the presence of the term surprise in economics and finance as well as how it is computed. Then, we present in Section 3 how emotions are defined in cognitive science, provide a formal definition of surprise, and describe the surprise process. Additionally, we present some theories regarding how artificial surprise can be computed. In Section 4 we describe a case study in which we compare the two different perspectives on surprise. In Section 5 we discuss some similarities and differences between the perspectives and point out some possible applications of the cognitive science perspective on artificial surprise.

2. Surprise in economics and finance

In this section we briefly present which investment and trading strategies are used by market participants, show evidence of the presence of the term surprise in the context of economics and finance as well as how it is computed.

2.1. Investment and trading strategies

In the scenario of a stock market there are, generally speaking, three investment or trading strategies commonly used by market participants in creating beliefs, expectations, and goals.

First, technical analysis (also referred to as graphical analysis) (e.g., [8]) make use of a myriad of statistical indicators derived from and build upon stocks data together with graphical patterns and tools to understand stock prices behaviour, identify trends and ultimately to discover and explore profit opportunities. The underlying assumption is that prices incorporate all relevant information, and both past behaviour and returns are rich in information concerning future behavior so that they can be used to some extent to predict or indicate future movements, i.e., history repeats itself.

Second, some market participants, known as "noise traders" [4], do not employ any "rational" mechanism, tend to form incorrect beliefs and expectations as well as base their trading strategies on what they consider to be worth information but actually is simple noise.

Third, fundamental analysis (e.g., [11]), whose rational is more in line with this work, is concerned with the estimation of the intrinsic (also referred to as fundamental or "fair") price. It begins with the estimation of the intrinsic value of a company, which includes both tangible and intangible assets. This task also involves the analysis of a wide range of factors like the understanding of the current macroeconomics scenario as well as the forecasting of possible scenarios. Additionally, it requires a close examination of financial information regarding the company (e.g., earnings). Finally, fundamental analysis is interested in identifying the internal and external risks which may affect the company. The work of fundamental analysis commonly results in a variety of ratios (e.g., P/E ratio, i.e., price-to-earnings ratio) which are used to estimate whether a current asset price deviates from the "fair" price. The problem relies on the fact that fundamentals are not totally observable. Forecasters often diverge in their opinions and forecasts to that the success of fundamental analysis resides in the estimation accuracy regarding the "fair" price.

2.2. The term surprise in economics and finance

There are a series of examples of the presence of the term surprise in the context of finance and economics (e.g., interest rates [12], surprise indexes [27], and overreaction [5]). However, some of the most prominent examples come from the research related to earnings surprise (e.g., [3, 7]). Considering the rational presented by the fundamental analysis, the basic interpretation is that when the actual earnings, released by companies with a given periodicity, are higher (lower) than the expected by the so-called market consensus, market participants should, according to Efficient Market Hypothesis (EMH), react to this new and surprising information accordingly.

The rational regarding earnings surprise can also be applied to other financial and economic indicators of different kinds (e.g., leading economic indicators). Generally speaking, a better than expected data about a relevant indicator (e.g., unemployment rate) may signal that the economy is behaving better than the expected and therefore the earnings of the companies will be also higher than the expected. Some indicators such as the Citigroup Economic Surprise Index (CESI) try to gauge that (see [27] for more details). To illustrate how different indicators relate to each other as well as to stress the importance of earnings/profits for asset behaviour and prices, we show in Figure 1 the evolution of the *S&P*500, i.e., an index that includes 500 companies in leading industries in the U.S. economy, corporate business profits before tax, and real gross domestic product (GDP).

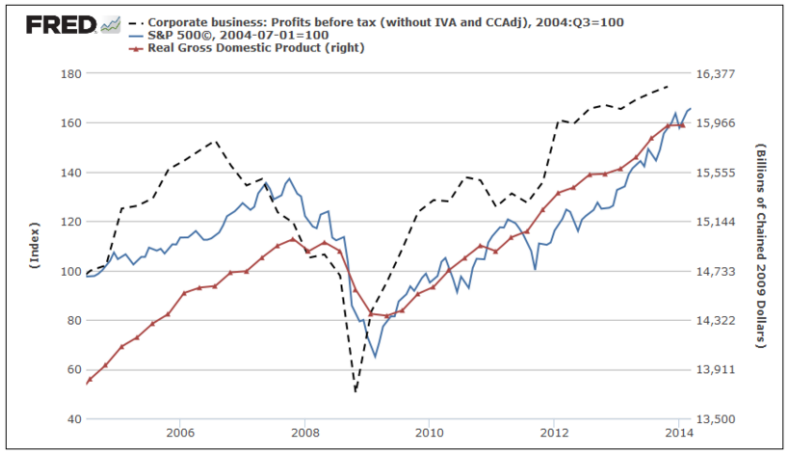


Figure 1. *S&P*500 index, corporate business profits before tax, and real gross domestic product (GDP), from 01-07-2004 to 01-07-2013, quarterly adjusted, generated with the Economic Research Federal Reserve Bank of St. Louis site - http://research.stlouisfed.org/.

2.3. Market consensus and the computation of surprise

Forecasts with the goal of creating a so-called market consensus are generated by a set of persons such as economists and professionals of financial markets. For instance, traditional economic and financial services periodically surveys groups of economists and professionals on a set of indicators. In this context, there are several different methods for computing earnings surprise (e.g., [14]). The earnings surprise is calculated by performing some kind of comparison between the actual earnings with the consensus estimate (generally the mean or the median of the forecasts).

The basic method for computing the unexpected earnings (UE_q) is calculated by dividing the actual earnings (EPS_q) by the consensus estimate (EST_q) in the time preceding the announcement, e.g., a quarter q: $UE_{q1} = EPS_q/EST_q$ (Equation 1). The unexpected earnings (UE_q) can also be calculated by a slightly different method: $UE_{q2} = EPS_q - EST_q$ (Equation 2). Also, there are two different and more sophisticated scaling methods for computing the unexpected earnings. The scaled unexpected earnings $(SCUE_q)$ is calculated by dividing unexpected earnings (UE_{q2}) by the absolute value of reported eps (EPS_q): $SCUE_q = \frac{UE_{q2}}{abs(EPS_q)}$ (Equation 3). Similarly, the standardized unexpected earnings (SUE_q) is calculated by dividing the unexpected earnings (UE_q) by the standard deviation of the consensus estimate $(\sigma\ EST_q)$: $SUE_q = \frac{UE_{q2}}{\sigma EST_q}$ (Equation 4). The same rational used to compute earnings surprise can be applied to compute the surprise "felt" by market participants with respect to other micro and macro economic and financial indicators.

3. Surprise in cognitive science

In this section we briefly present how emotions are defined by cognitive emotion theories. Then, we provide a formal definition of surprise, describe the surprise process, as well as present some theories regarding artificial surprise.

3.1. Cognitive Emotion Theories

Cognitive emotion theories (e.g., [26]), such as the Belief-Desire Theory of Emotions (BDTE) [25], rely on the assumption that emotions are mental states elicited as a result of the evaluation or appraisal of stimuli of all kinds (e.g., events) and can be computed in terms of cognitions (beliefs) and motives (desires). Beliefs are mental states in which one holds a particular proposition to be true, whereas desires represent the motives or future states that one wants to accomplish.

The BDTE [25] is consisted of propositions, beliefs, desires, new beliefs, and two hard-wired comparator mechanisms, namely the Belief-Belief Comparator (BBC) and the Belief-Desire Comparator (BDC). The conceptual framework of the BDTE is the same as the belief-desire theory of action which inspired the BDI (belief-desire-intention) approach to artificial agents. In this section we provide a concise but sufficient description of the BDTE relevant to this work. To understand the nature of both the BDTE and the BDI approach, please see Reisenzein et al. [26].

A proposition p is represented as a tuple $\langle S, B, D \rangle$ where S is the mental language expressing the proposition p, B and D are quantities representing, respectively, the agent's degree of belief and desire regarding proposition p. The strength of a belief in a proposition p at time t, is defined as $b(p,t)$, where $b(p,t) \in \mathbb{R}$ and $0.0 \leq b(p,t) \leq 1.0$, where 1.0 denotes certainty that p, 0.5 maximal uncertainty, and 0.0 certainty that not p. Similarly, the strength of a desire about a proposition p at time t, is defined as $d(p,t)$, where $d(p,t) \in \mathbb{Z}$ and $-100 \leq d(p,t) \leq +100$, where positive values denote desire in favor of p, negative values denote desire against p, and 0 denotes indifference. A new belief is the belief or fact in a proposition that agents receive basically through its sensors (e.g., vision).

The Belief-Belief Comparator (BBC) compares each newly acquired belief with all pre-existing beliefs, looking for match versus mismatch. A match (mismatch) means that a pre-existing belief was confirmed (disconfirmed) by the newly acquired belief. As a result, BBC yields either a belief-confirmation signal or belief-disconfirmation signal. Similarly, the Belief-Desire Comparator (BDC) compares each newly acquired belief with all pre-existing desires, looking for match versus mismatch. A match (mismatch) means that a desire was "fulfilled" ("frustrated"). As a result, BDC yields either a desire-fulfillment signal or desire-frustration signal. BDTE defines emotions as products or signals produced by the BBC and BDC.

For example, suppose an agent has $b(p,t) = 0.9$ and $d(p,t) = +80$, i.e., at time t the agent believes proposition p will happen, and has a desire in favor of p. When the agent receives a new belief at time $t+1$ that actually p not happened, the BBC yields a belief-disconfirmation signal (surprise), since what the agent believed at t as less likely really happened. Similarly, the BDC yields a desire-frustration signal, since the agent has a desire in favor of p.

3.2. Surprise

Surprise is a neutral valence emotion, formally defined as a peculiar state of mind, usually of brief duration, caused by unexpected events, or proximally the detection of a contradiction or conflict between newly acquired and pre-existing beliefs [23, 19]. Surprise serves us in many functions such as attention and learning, being considered, from an evolutionary perspective, crucial for survival in a rapidly changing environment.

Surprise is closely related to how beliefs are stored in memory. Our semantic memory, i.e., our general knowledge and concepts about the world, is assumed to be represented in memory through knowledge structures known as schemas (e.g., [1]). A schema is a well-integrated chunk of knowledge or sets of beliefs, which main source of information available comes from abstraction from repeated personally experienced events or generalizations. Schemas serve the interpretation of present and past, and make it possible the prediction of future events by means of the adaptive guidance of action.

3.2.1. The Surprise Process

Meyer et el. [23] proposed a cognitive-psychoevolutionary model of surprise. They claim surprise-eliciting events elicit a four-step sequence of processes.

The first step is the appraisal of an event as unexpected or schema-discrepant. For instance, in the case of the BDTE, one of the functions of the BBC is the detection of disconfirmation between pre-existing and newly acquired beliefs or, in other words, to detect whether a schema-discrepancy occurs.

If the degree of unexpectedness or schema-discrepancy exceeds a certain threshold then, in the second step, surprise is experienced, ongoing mental process are interrupted and resources such as attention are reallocated towards the unexpected event.

The third step is the analysis and evaluation of the unexpected event. It generally includes a set of subprocesses namely the verification of the schema discrepancy, the analysis of the causes of the unexpected event, the evaluation of the unexpected event's significance for well-being, and the assessment of the event's relevance for ongoing action. It is assumed that some aspects of the analysis concerning the unexpected event are stored as part of the schema for this event so that in the future analysis of similar events can be significantly reduced both in terms of time and cognitive effort.

The fourth step is the schema update. It involves producing the immediate reactions to the unexpected event (if it is the case), and/or operations such as the update, extension, or revision of the schema or sets of beliefs that gave rise to the discrepancy. The schema change (belief update process) ideally enables one to some extent to predict and control future occurrences of the schema-discrepant event and, if possible, to avoid the event if it is negative and uncontrollable, or to ignore the event if it is irrelevant for action.

3.2.2. Artificial Surprise

Two models of artificial surprise for artificial agents can be stressed namely the model proposed by Macedo and Cardoso [20] and the model proposed by Lorini and Castelfranchi [18]. Both models were mainly inspired by the cognitive-psychoevolutionary model of surprise proposed by Meyer et el. and have influence of the analysis of the cognitive causes of surprise from a cognitive science perspective proposed by Ortony and Partridge [24]. To a detailed description of the similarities and differences of the models see [19]. There are other approaches of artificial surprise proposed for other context rather than artificial agents (e.g., [13]). We consider the model proposed by Macedo and Cardoso as the simplest, easy to understand, and straightforward existing model for artificial surprise.

Macedo et el. carried out an empirical study [20] with the goal of investigating how to compute the intensity of surprise in an artificial agent. They proposed several alternative functions for computing the surprise intensity based on the assumption that the

surprise "felt" by an agent elicited by an event E_g is proportional to the degree of unexpectedness of the event E_g. They examined the functions by carrying out a two-step experiment. First, they collected ratings of probability and surprise intensity provided by human participants in two domains (political elections and sports games). Second, they empowered artificial agents with the alternative functions as well as with the ratings of probability provided by human participants so that the artificial agents were able to compute the surprise intensity values. Finally, the values obtained by the artificial agents were compared with the actual surprise intensity given by human participants.

This study suggested that the intensity of surprise about an event E_g, from a set of mutually exclusive events E_1, E_2, ..., E_m, is a nonlinear function of the difference, or contrast, between its probability/belief and the probability/belief of the highest expected event (E_h) in the set of mutually exclusive events E_1, E_2, ..., E_m. Formally, let (Ω, A, P) be a probability space where Ω is the sample space (i.e., the set of possible outcomes of the event), $A = A_1, A_2, ..., A_n$, is a $\sigma field$ of subsets of Ω (also called the event space, i.e., all the possible events), and P is a probability measure which assigns a real number $P(F)$ to every member F of the $\sigma field$ A. Let $E = E_1, E_2, ..., E_m$, $E_i \in A$, be a set of mutually exclusive events in that probability space with probabilities $P(E_i) \geq 0$, such that $\sum_{i=1}^{m} P(E_i) = 1$. Let E_h be the highest expected event from E. The intensity of surprise about an event E_g, defined as $S(E_g)$, is calculated as: $S(E_g) = log_2(1 + P(E_h) - P(E_g))$ (Equation 5). In each set of mutually exclusive events, there is always at least one event whose occurrence is unsurprising, namely E_h.

4. Comparing different perspectives on surprise: a case study

We carried out a case study to compare the computation of surprise from the perspective of economics and finance to the perspective of cognitive science. We obtained the free data related to the forecasts from the Wall Street Journal (WSJ) (http://projects.wsj.com/econforecast/). The WSJ monthly surveys a group of nearly 50 economists on more than 10 major economic indicators. For this case study we selected the unemployment rate as indicator as well as compiled its data from 01-07-2008 to 01-01-2014 in a total of 70 months. Similarly, we obtained the actual/released data regarding the selected economic indicator from the Economic Research Federal Reserve Bank of St. Louis (http://research.stlouisfed.org/).

The economical and financial approach is straightforward. We applied the earnings surprise rational to compute the surprise regarding the unemployment rate (UR). We computed the mean and the standard deviation σEST_m of the forecasts, where $_m$ refers to the monthly periodicity, the unexpected unemployment rate by UUR_m (Equation 1) and UUR_m (Equation 2), the scaled unexpected unemployment rate $SCUUR_m$ (Equation 3) and the standardized unexpected unemployment rate $SUURE_m$ (Equation 4).

The results of this case study are shown in Figure 2.

5. Discussion and conclusion

First of all, it is important to bear in mind that our goal is not to provide evidence neither in favor of nor against the use of a given indicator. We could have selected another indicator(s) without any impact on our results.

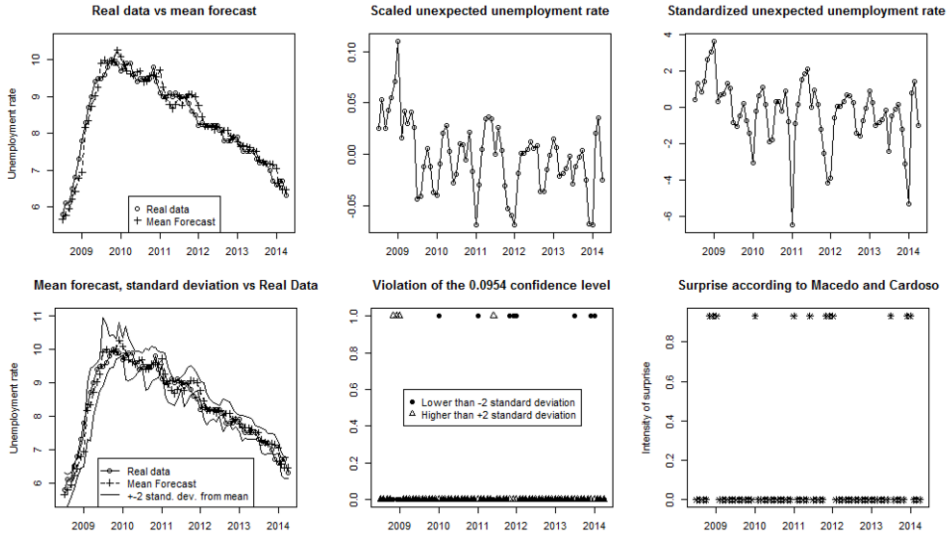


Figure 2. Top: Real data vs mean forecast (left), $SCUUR_m$ (Equation 3) (center), (Equation 4), $SUURE_m$ (Equation 4) (right); Bottom: Mean forecast, standard deviation vs real data (left), Violation of confidence level (center), $S(E_g)$ (Equation 5) (right).

The cognitive science perspective rely on the subjective probabilities to compute artificial surprise (Equation 5). As it should be, considering the underlying theory, it does not capture whether it is positive or negative. Additionally, the higher the difference between the outcomes, the higher may be the surprise.

The economical and financial perspective can be thought of as a pure mathematical approach. It uses a standard deviation to try to capture whether forecasters diverge or converge as well as to present whether the surprise is positive or negative, however, it requires an understanding of the related indicator (e.g., a lower than expected data in unemployment rate is a good thing).

Unlike the straightforward economical and financial perspective, the cognitive science perspective requires the creation of the outcomes and the estimation of its subjective probabilities. For instance, in this work we rely on the assumption that forecasts follow a Normal distribution pattern. We could have used a normality test (e.g., Shapiro Wilk test) or analyzed other variables (e.g., kurtosis) to test that assumption. However, the results may be true for one group and false for another group (the efforts spent on such task would be probably in vain). Nevertheless, we consider that relying on this assumption does not have any impact on our results, especially taking into account that we used this rational just to create the so-called consensus. Despite that, one should indeed be careful in assuming the Normal distribution pattern in other contexts (e.g., there is substantial evidence that financial asset returns are not normally distributed [16]). Other methods would ideally include mechanisms such as self-reports so that forecasters should be able to express their confidence in their forecasts. With this information one may be able to weight the forecasts more precisely. Another approach may include the analysis of different financial instruments (e.g., options) to try to infer the beliefs of market participants by observing their actions.

Considering the cognitive science perspective, what we are able to say is that the release of an indicator may elicit surprise only on a given set of persons, the so-called market consensus and not the entire market. Instead of carrying meaningful information, terms such as "beat (miss) market expectation" seem to carry just "noise" and, in the end, does not seem to make much sense, at least from this perspective.

The findings of the behavioral economics together with empirical observations of the behaviour of market participants have been stressing the necessity of adopting novel approaches [10] in order to improve our understanding of complex economical and financial systems [6]. One of the possible ideas is the use of cognitive modeling approaches. Having cognitive agents means that artificial agents will be empowered with mechanisms similar to or inspired in those used by humans. Therefore, the behaviour of artificial agents tends to be closer to the behaviour of humans in a similar scenario. Our multidisciplinary work is line with this context. It is, as far as we know, one of the first attempts (possibly the first) to bring together two different approaches on surprise, by presenting and describing what surprise is from the cognitive science perspective and, furthermore, how an artificial surprise model can be computed and applied to economics and finance. The use of cognitive modeling approaches in this kind of complex systems is in its early stages. We consider that the use of relatively simple but powerful tools in conjunction with other cognitive models, like those discussed in this work, offers a rich and novel set of possibilities for investigation and to shed light on the behaviour of cognitive agents.

References

[1] Alan Baddeley, Michael Eysenck, and Michael C. Anderson. *Memory*. Psychology Press, 1 edition, February 2009.

[2] Eli Bartov, Dan Givoly, and Carla Hayn. The rewards to meeting or beating earnings expectations. *Journal of Accounting and Economics*, 33(2):173–204, June 2002.

[3] Victor L. Bernard and Jacob K. Thomas. Post-earnings-announcement drift: Delayed price response or risk premium? *Journal of Accounting Research*, 27:1–36, 1989. ArticleType: primary_article / Issue Title: Current Studies on The Information Content of Accounting Earnings / Full publication date: 1989 / Copyright 1989 Accounting Research Center, Booth School of Business, University of Chicago.

[4] Fischer Black. Noise. *Journal of Finance*, 41(3):529–543, 1986.

[5] Werner F M De Bondt and Richard Thaler. Does the stock market overreact? *Journal of Finance*, 40(3):793–805, 1985.

[6] Jean-Philippe Bouchaud. Economics needs a scientific revolution. *Nature*, 455(7217):1181, October 2008.

[7] Lawrence D. Brown. Small negative surprises: frequency and consequence. *International Journal of Forecasting*, 19(1):149–159, January 2003.

[8] Robert Edwards and John Magee. *Technical Analysis of Stock Trends*. Snowball Publishing, July 2010.

[9] Eugene F. Fama. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2):383–417, May 1970.

[10] J. Doyne Farmer and Duncan Foley. The economy needs agent-based modelling. *Nature*, 460(7256):685–686, 2009.

[11] Robert G. Hagstrom. *The Warren Buffett Way*. Wiley, 2 edition, October 2005.

[12] Young Wook Han. The effects of US macroeconomic surprises on the intraday movements of foreign exchange rates: Cases of USD-EUR and USD-JPY exchange rates. *International Economic Journal*, 24(3):375–396, 2010.

[13] Laurent Itti and Pierre Baldi. Bayesian surprise attracts human attention. *Vision Research*, 49(10):1295–1306, June 2009.

[14] Michael Kaestner. Investors' misreaction to unexpected earnings: Evidence of simultaneous overreaction and underreaction. *Behavioral Finance*, 3, 2006.

[15] Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2):263–291, 1979.

[16] Andrew Lo and Mark Mueller. WARNING: physics envy may be hazardous to your wealth! *Journal of Investment Management*, 8:13–63, 2010.

[17] Andrew W. Lo. Reconciling efficient markets with behavioral finance: The adaptive markets hypothesis. *Journal of Investment Consulting*, 7:21–44, 2005.

[18] Emiliano Lorini and Cristiano Castelfranchi. The cognitive structure of surprise: looking for basic principles. *Topoi: An International Review of Philosophy*, 26:133–149, 2007.

[19] Luis Macedo, Amilcar Cardoso, Rainer Reisenzein, Emiliano Lorini, and C. Castelfranchi. Artificial surprise. In *Handbook of Research on Synthetic Emotions and Sociable Robotics: New Applications in Affective Computing and Artificial Intelligence*, pages 267–291. 2009.

[20] Luis Macedo, Rainer Reisenzein, and Amilcar Cardoso. Modeling forms of surprise in artificial agents: empirical and theoretical study of surprise functions. In *26th Annual Conference of the Cognitive Science Society*, pages 588–593, 2004.

[21] Burton G Malkiel. The efficient market hypothesis and its critics. *Journal of Economic Perspectives*, 17(1):59–82, 2003.

[22] Harry Markowitz. Portfolio selection. *Journal of Finance*, 7:77–91, 1952.

[23] W. U. Meyer, Rainer Reisenzein, and A. Schutzwohl. Toward a process analysis of emotions: The case of surprise. In *Motivation and Emotion*, volume 21, pages 251–274, 1997.

[24] Andrew Ortony and Derek Partridge. Surprisingness and expectation failure: what's the difference? In *Proceedings of the 10th international joint conference on Artificial intelligence - Volume 1*, pages 106–108, Milan, Italy, 1987. Morgan Kaufmann Publishers Inc.

[25] Rainer Reisenzein. Emotions as metarepresentational states of mind: Naturalizing the belief-desire theory of emotion. *Cognitive Systems Research*, 10(1):6–20, March 2009.

[26] Rainer Reisenzein, Eva Hudlicka, Mehdi Dastani, Jonathan Gratch, Koen Hindriks, Emiliano Lorini, and John-Jules Meyer. Computational modeling of emotion: Towards improving the inter- and intradisciplinary exchange. *IEEE Transactions on Affective Computing*, 99(1):1, 2013.

[27] Chiara Scotti. Surprise and uncertainty indexes: real-time aggregation of real-activity macro surprises. International Finance Discussion Paper 1093, Board of Governors of the Federal Reserve System (U.S.), 2013.

[28] Herbert A. Simon. A behavioral model of rational choice. *The Quarterly Journal of Economics*, 69(1):99–118, February 1955.

[29] Karl-Erik Warneryd. *Stock-Market Psychology: How People Value and Trade Stocks*. Edward Elgar Publishing, illustrated edition edition, October 2001.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-41

Temporal Plan Quality Improvement and Repair using Local Search

Josef Bajada [1], Maria Fox and Derek Long
Department of Informatics, King's College London,
Strand, London WC2R 2LS, United Kingdom.
{josef.bajada, maria.fox, derek.long}@kcl.ac.uk

Abstract. This paper presents an approach to repair or improve the quality of plans which make use of temporal and numeric constructs. While current state-of-the-art temporal planners are biased towards minimising makespan, the focus of this approach is to maximise plan quality. Local search is used to explore the neighbourhood of an input seed plan and find valid plans of a better quality with respect to the specified cost function. Experiments show that this algorithm is effective to improve plans generated by other planners, or to perform plan repair when the problem definition changes during the execution of a plan.

Keywords. temporal planning, scheduling, optimisation, local search, plan repair

Introduction

Real world planning problems often need to take into account time and resources together with concurrency and exogenous events. PDDL 2.1 [1] and 2.2 [2] introduced the constructs necessary to model such problem domains, under the form of durative actions, numeric fluents and timed initial literals. Nevertheless, most of the state-of-the-art temporal planners struggle to cope with complex metric functions and concurrency. Most planners are biased to generate feasible plans that minimise plan length or makespan. However, in some problem domains it is preferable to generate plans that minimise a certain cost rather than plan duration. This is especially true for domains where plan execution is continual and the goal is to maintain some variables within specific bounds, or new goals are queued into the system during the plan's execution. One example is the demand-side electricity aggregator domain. In this case the system needs to find a plan, comprising of task-completing actions and load-shifting actions, within a planning horizon that features frequent electricity price fluctuations, with the objective of minimising wholesale electricity costs [3].

In this paper we present a domain-independent approach that generates high quality plans, in terms of some cost function. The proposed technique involves performing a local search on a provided input seed plan to explore its neighbourhood for better quality plans. This process can then be repeated until the time allocated for the algorithm has

[1]This research is funded by the UK Engineering and Physical Sciences Research Council (EPSRC) as part of the project entitled *The Autonomic Power System* (Grant Ref: EP/I031650/1)

elapsed. One can also utilise one of the existent temporal planners to generate a valid feasible plan, and use that as the input seed plan to find better quality plans. The proposed algorithm is also effective for plan repair, when exogenous events or changes in goals invalidate a plan.

1. Background

Local search is commonly used in various combinatorial problems, such as discrete optimisation, and it has also proved to be successful in classical planning. LPG [4] is one popular planner that uses this approach. An action graph that connects the initial state to the goal is found by adding or removing random actions from the problem's planning graph. The process is then repeated until the action graph becomes a solution graph, that is, until it has no flaws and the goal facts are present in the final state, making it a valid plan. While the original version of LPG caters only for propositional planning, it was later enhanced [5] to support some of the temporal constructs introduced in PDDL 2.1. Local search was also proposed as a solution for plan improvement in the context of classical planning [6]. In this case a neighbourhood graph of states is constructed from the states of the given seed plan, and the shortest plan that leads from the initial state to the goal is then extracted using Dijkstra's algorithm.

We propose to use local search to find plans in domains that not only require concurrent durative actions, but also have invariant conditions that are potentially mutually exclusive. Furthermore, our objective is to find plans of a high quality with respect to some metric. A problem's time-related constraints and characteristics, such as action durations and timed events, are used to build a planning time line. The respective plan violations at each time point are analysed and the respective metrics at each state are also calculated. This enables the algorithm to consider concurrent actions and also account for non-linear numeric effects. Most state-of-the-art temporal planners struggle with concurrent durative actions and non-linear numeric effects. While these planners are biased towards finding shorter plans, our approach can find plans of a better quality. Moreover, valid plans generated by these planners can be used as input seed plans for our algorithm, which will then search for a better plan in terms of some objective function. The input seed plan does not have to be valid, which also makes this algorithm useful for plan repair. If a plan becomes invalid due to changes in the environment, the new problem definition can be analysed in conjunction with the old plan to generate a new valid plan for the new version of the problem.

Local search algorithms comprise of two main components:

1. A **neighbourhood function**, which transforms an input state into a set of new but very similar states, with a very limited number of differences.
2. An **evaluation function**, which determines the states that are more desirable, according to some objective.

Using these two components new neighbours are generated, evaluated and explored according to the algorithm being used. Hill climbing algorithms incrementally choose neighbours that provide a better solution until no further improvements are possible. This carries the risk of getting stuck into a local minimum. Other flavours of local search algorithms include simulated annealing [7], which allows the exploration of inferior solutions with a certain probability, improving the chances of finding a global minimum.

2. Temporal Planning

Our notion of temporal planning follows the semantics of PDDL 2.1 [1], where actions have a duration, together with conditions and effects that are associated with the start, end or execution of the action. We also consider exogenous timed events, as defined in PDDL 2.2 [2]. The following definitions formulate the fundamental underpinnings of these semantics.

Definition 2.1. A temporal problem, $P = \langle A, I, L, G \rangle$, consists of a set of possible actions A, the initial state I, a set of timed initial literals L, and a goal condition G.

Definition 2.2. A temporal plan, $\pi = \{a_0, a_1, ..., a_k\}$, is defined as a list of durative actions. Each action a has a start time, denoted $start(a)$, and a duration, $dur(a)$.

The start time of a durative action is a positive rational number, indicating the time, after the start of the plan, when the action should commence. The duration is also a positive rational number. Multiple durative actions can be executed concurrently in a temporal plan.

Definition 2.3. A durative action, a, may have conditions that need to be satisfied just before it starts, referred to as $startCond(a)$, conditions that need to be satisfied just before it ends, referred to as $endCond(a)$, and invariant conditions that need to hold throughout the execution of the action, referred to as $inv(a)$.

Definition 2.4. A durative action, a, may have effects that are applied when the action starts, referred to as $startEff(a)$, and effects that are applied when the action ends, referred to as $endEff(a)$.

PDDL 2.1 [1] also defines continuous effects, to represent continuously changing values with respect to the time elapsed from the start of the action. This construct is not currently supported in the work presented here.

Timed initial literals (TILs) were introduced in PDDL 2.2 [2] to support predictable exogenous state-changing events that will occur at some predetermined time during the plan. A timed initial literal, l, has a time when it is predicted to take place, $time(l)$, and an associated proposition that will become *true* or *false*. This construct is useful to denote external changes in the environment, or time windows when certain activities can take place. For example, in the demand-side electricity aggregator domain, TILs are used to perform tariff switches. TILs can be seen as instantaneous actions without any preconditions that will take place at a predefined time.

Each durative action, a, can be translated into two *snap actions* [8], $a_{\vdash}$, corresponding to the start of the action, and $a_{\dashv}$, corresponding to the end of the action. Snap actions are essentially instantaneous actions where $pre(a_{\vdash}) = startCond(a)$, $pre(a_{\dashv}) = endCond(a)$, $eff(a_{\vdash}) = startEff(a)$ and $eff(a_{\dashv}) = endEff(a)$.

In order to avoid ambiguity in the application of action effects, a total order is enforced by introducing a minimal time separation ε between two successive actions, and only one snap action is allowed to be applied at a certain point in time. The sequence $E_\pi = \{e_1, e_2, ..., e_n\}$ corresponds to the state-changing activities (snap actions and TILs) of π, with $time(e_i)$ denoting the time when e_i will be executed.

Definition 2.5. A plan's state time-line $\Upsilon(\pi) = \{\langle 0, s_0\rangle, \langle t_1, s_1\rangle, \langle t_2, s_2\rangle, ..., \langle t_n, s_n\rangle\}$ is a sequence of pairs $\langle t, s\rangle$ where t is a rational number corresponding to the time from the start of the plan when the current state will become s. State s_0 is the initial state.

A period, p_i, denotes the time interval between two successive states on the time-line. These periods are used to identify possible insertion points where new actions could be added to the plan. The last period p_n is open-ended since it denotes the time interval that follows the final state s_n, thus allowing actions to be added to the end of the plan.

$$\forall \langle t_i, s_i\rangle \in \Upsilon(\pi), \text{ where } 0 \leq i \leq n : p_i = \begin{cases} \langle t_i, t_{i+1}\rangle, & \text{if } i < n \\ \langle t_i, \infty\rangle, & \text{if } i = n \end{cases} \tag{1}$$

The invariant conditions of a period p_i correspond to all the invariant conditions of the actions running concurrently throughout that period, as defined in Equation 2.

$$\begin{aligned} invp(p_i) &= \bigcup_{a \in A_i} inv(a), \text{ where } 0 \leq i < n \text{ and} \\ A_i &= \{a | start(a) \leq t_i < t_{i+1} \leq start(a) + dur(a)\} \end{aligned} \tag{2}$$

3. The Neighbourhood of a Temporal Plan

We define a plan π' as the neighbour of a plan π, denoted $\pi' \in N(\pi)$, if π' can be obtained by either *adding* one applicable action at some point on the time-line, *removing* an existent action from the plan, or *moving* an action to start and end at a different time point on the plan's time-line. Each of these operations will naturally change the plan's state time-line and also the invariant conditions for each period.

3.1. Adding an Action

By adding a new durative action, a_{new}, to start within period p_i and end within period p_j, where $0 \leq i \leq j \leq n$, two new states, s_s and s_e, will be added to the time-line, corresponding to the two snap actions of a_{new}. The state $s_s = startEff(a_{new})(s_i)$, reflects the start effects of a_{new} applied to state s_i. The state $s_e = endEff(a_{new})(s'_j)$, reflects the application of the end effects of the action, where s'_j is the new state obtained from applying all subsequent actions following s_s in sequence up till t_j. All the states s_x, where $i < x \leq n$, following s_s on the time-line, need to be propagated and updated to s'_x, to account for the effects of the new action. All the states $\{s_0, ..., s_i\}$ will remain the same while the subsequent states will be updated. The new action will naturally run concurrently with any other actions scheduled to run during periods $\{p_i, ..., p_j\}$, making it possible to find solutions in cases where concurrency is required [9].

A durative action, a_{new}, is only eligible for addition to the plan's time-line at an arbitrary time point during period p_i if it satisfies the following compatibility criteria:

1. The state s_i satisfies the start conditions of a_{new}, that is $s_i \models startCond(a_{new})$.

2. The new state s_s satisfies all the invariant conditions during that period, including those of a_{new}, that is $s_s \models inv(a_{new}) \cup invp(p_i)$.
3. All the subsequent states of s_s up to and including s_e satisfy the invariant conditions of a_{new}, that is $\forall s \in \{s_s, s'_{i+1}, ..., s'_j, s_e\} : s \models inv(a_{new})$

This ensures that the new action is compliant with the plan. However, this does not mean that subsequent conditions or invariants associated with any periods p_j, where $j > i$, will not be violated by adding a_{new} at p_i. Nevertheless, invalid neighbouring plans can still lead to valid plans that are further away in the neighbourhood of the seed plan π. By considering a violated plan π', that can be repaired through further exploration, the algorithm improves its chances of escaping from local minima.

The set $\{p_j | i \leq j \wedge start(p_j) < end(p_i) + dur(a_{new}) \wedge end(p_j) > start(p_i) + dur(a_{new})\}$ (where $start(p)$ and $end(p)$ correspond to the start and end time-point of period p respectively) defines the possible candidate periods where a_{new} can end. Each possibility that also satisfies the above compatibility criteria can be used to obtain a valid neighbouring plan π'. The start time of a_{new} is then set to an arbitrary value that satisfies $max[start(p_i), start(p_j) - dur(a_{new})] < start(a_{new}) < min[end(p_i), end(p_j) - dur(a_{new})]$ and $\forall e \in E_\pi : start(a_{new}) \neq time(e) \neq start(a_{new}) + dur(a_{new})$.

3.2. Removing an Action

Any durative action, a_{del}, that is already in the plan, can be selected for removal. The two states, s_d and s_r, where $0 < d < r \leq n$, correspond to states obtained by applying the start and end effects of the action a_{del} respectively. By removing the action, these two states are removed from the plan's time-line, and the rest of the states that follow s_d are updated accordingly.

3.3. Moving an Action

Moving an action, a_{mov}, can be seen as a macro action that involves removing an action and adding it again at a different point on the time-line. The effect of this modification would be a change in the plan's total ordering of the actions rather than a change of the plan's set of actions. This is especially useful in domains where the order of the actions has an impact on the cost of the plan, or exogenous events change the action costs at specific time points. This move needs to satisfy the same compatibility criteria used for adding an action to be considered a valid operation.

Substituting an action with another one might intuitively also seem like a valid neighbourhood operation. However, the benefits of such an operation in a temporal context, where actions have different durations and action swapping can change the total ordering of state-changing events, need to be analysed further.

4. Evaluation of Neighbouring Temporal Plans

A temporal plan obtained through the neighbourhood function needs to be evaluated on two levels. Firstly, we need to determine how close the plan is to a valid solution. Secondly, we need to measure the cost of the plan, with respect to some cost function. In order not to get stuck in local minima, inferior plans to the current one are also evaluated

and explored, with a certain probability. This is governed by a probability distribution that diminishes proportionally with the exploration *distance*, that is the number of nodes traversed from the best one. This means that immediate neighbours of the best plan have a higher probability of being accepted than ones further away in the neighbourhood graph. However, valid plans that have a better cost than the current best plan will always be accepted. This is intuitively similar to the approach used in simulated annealing [7], where a *temperature* value decreases with each iteration, and the probability of accepting a weaker solution is computed using a function of this *temperature*. However, in our case we reset the *distance* each time we restart searching again from the best plan, thus assigning a high acceptance probability to closer neighbours, irrespective of when they were discovered.

4.1. Computing a Plan's Validity

Let $\alpha_i = \{a|start(a) = t_i\}$ be the set of actions in a plan π starting at a time point t_i, and $\omega_i = \{a|start(a) + dur(a) = t_i\}$ be the set of actions ending at a time point t_i, where $0 < i \leq n$. Equation 3c defines the set of conditions that need to be satisfied at time point t_i, in terms of the start and end conditions of actions starting or ending at t_i, defined by Equations 3a and 3b respectively. $G(t_i)$ represents any goal conditions that need to be satisfied at t_i. Goals that are only required to be satisfied at the end of the plan and do not have any time constraints are included in the set $G(t_n)$.

$$startCondT(t_i) = \bigcup_{a \in \alpha_i} startCond(a) \tag{3a}$$

$$endCondT(t_i) = \bigcup_{a \in \omega_i} endCond(a) \tag{3b}$$

$$cond(t_i) = startCondT(t_i) \cup endCondT(t_i) \cup invp(p_i) \cup G(t_i) \tag{3c}$$

The conditions that are actually satisfied at t_i are those that are satisfied by the state s_{i-1}, defined as $\sigma(t_i) = \{c|c \in cond(t_i) \wedge s_{i-1} \models c\}$. Conversely, $\phi(t_i) = cond(t_i) \setminus \sigma(t_i)$ defines the set of conditions that are not satisfied at t_i. A plan is considered valid if $\forall t_i \in \{t_1, ..., t_n\} : \phi(t_i) = \emptyset$. If a plan is invalid, the number of violations $v(\pi)$ is calculated by accumulating all the unsatisfied conditions at a given time point t_i, excluding any conditions that were already unsatisfied at t_{i-1}. This helps to avoid inflating the violation count from a common condition that is required by more than one action or by the same action at more than one point on the time-line. This process is defined recursively through Equations 4a to 4d, where $0 < i \leq n$. $cond^+(t_i)$ represents the cumulative set of conditions that need to be satisfied at t_i together with any conditions that were not satisfied at t_{i-1}. Similarly, $\sigma^+(t_i)$ represents the cumulative conditions that are satisfied at t_i, including any conditions carried forward from previous states, and conversely $\phi^+(t_i)$ represents the cumulative set of conditions unsatisfied at t_i. We can then extract $\phi^*(t_i)$, the set of conditions that are introduced at t_i.

$$cond^{+}(t_i) = cond(t_i) \cup \phi^{+}(t_{i-1}) \tag{4a}$$

$$\sigma^{+}(t_i) = \{c | c \in cond^{+}(t_i) \wedge s_{i-1} \models c\} \tag{4b}$$

$$\phi^{+}(t_i) = cond^{+}(t_i) \setminus \sigma^{+}(t_i) \tag{4c}$$

$$\phi^{*}(t_i) = \phi(t_i) \setminus \phi^{+}(t_{i-1}) \tag{4d}$$

If a condition becomes satisfied at t_j, but becomes unsatisfied again at a later time t_k, (where $i < j < k$), and is needed by an action at or after t_k, it is counted twice. This is because at least two additional actions are needed to make the plan valid. Equation 5 defines the violation count $v(\pi)$ for a plan π. If $v(\pi) = 0$, the plan π is valid.

$$v(\pi) = \sum_{i=1}^{n} |\phi^{*}(t_i)| \tag{5}$$

4.2. Acceptance Probability Function

In order to promote the exploration of a broader neighbourhood we need a function that has a high probability of accepting plans that are close neighbours of the initial plan. On the other hand, in order to avoid exploring deep branches that do not lead to a solution, we need such a function to asymptotically decrease towards 0 in proportion to the exploration distance d, parametrised by the maximum distance m we want to explore. One candidate that fits these criteria is the sigmoid function defined in Equation 6.

$$g_m(d) = 1 - \left(\frac{1}{1 + e^{(m-d)}} \right) \tag{6}$$

We also want to increase the chances of accepting a good candidate at any exploration distance, depending on its *fitness* ratio with respect to the previous plan. This value is computed using the function $f(\pi', \pi)$, which compares two plans π' and π, as shown in Equation 7.

$$f(\pi', \pi) = \begin{cases} \dfrac{c(\pi') + 1}{c(\pi) + 1}, & \text{if } v(\pi) = v(\pi') = 0 \\ \dfrac{v(\pi') + 1}{v(\pi) + 1}, & \text{otherwise} \end{cases} \tag{7}$$

If both plans are valid, the actual cost function $c(\pi)$ is used, which is subject to the respective domain and problem instance. If one of the plans is invalid, the fitness ratio with respect to an adjacent plan is calculated using the number of violations in the two plans. In both cases 1 is added as a smoothing parameter. Equation 8 defines the acceptance probability function of exploring π' from its adjacent neighbouring plan π, with the current distance from the best plan π^* being d.

$$p_m(\pi', \pi, d) = g_m(d)^{f(\pi', \pi)} \tag{8}$$

5. Searching for Temporal Plans

The algorithm to search for a better temporal plan, starting from an initial seed plan π, involves exploring its neighbourhood until a better one is found. A plan π' is considered better than π, denoted by the relation $\prec_c$, if the plan π' has no violations, and either π is not a valid plan, or the cost of π is more than that of π'. Formally, $\pi' \prec_c \pi$ if $(v(\pi') = 0) \wedge (v(\pi) > 0 \vee c(\pi') < c(\pi))$. Given that the neighbourhood of a plan can be very large, a neighbour is generated randomly, with the search taking the form of a random walk guided by the acceptance probability function, p_m, as described in Algorithm 1. The set *visited* keeps track of the plans explored during one random walk to avoid cycles.

Algorithm 1 Local search for a better temporal plan

Require: Seed plan π^*, maximum distance m
1: $\pi \leftarrow \pi^*$; $d \leftarrow 0$
2: $visited \leftarrow \{\pi\}$
3: **while** not($\pi \prec_c \pi^*$) and not($term$) **do**
4: **if** $d \geq m$ **then**
5: $\pi \leftarrow \pi^*$; $d \leftarrow 0$
6: $visited \leftarrow \{\pi\}$
7: **end if**
8: $\pi' \leftarrow$ select random plan from $N(\pi) \setminus visited$
9: **if** $(v(\pi') > 0)$ and $(v(\pi') < v(\pi^*))$ **then**
10: $\pi^* \leftarrow \pi \leftarrow \pi'$; $d \leftarrow 0$
11: $visited \leftarrow \{\pi\}$
12: **else if** $\pi' \prec_c \pi$ **then**
13: $\pi \leftarrow \pi'$; $d \leftarrow d + 1$
14: $visited \leftarrow visited \cup \{\pi\}$
15: **else**
16: $r \leftarrow$ random double between 0 and 1
17: **if** $r \leq p_m(\pi', \pi, d)$ **then**
18: $\pi \leftarrow \pi'$; $d \leftarrow d + 1$
19: $visited \leftarrow visited \cup \{\pi\}$
20: **end if**
21: **end if**
22: **end while**
23: **return** π

In its simplest form, this algorithm carries the risk of running indefinitely if no better plan is found, or if no valid solution actually exists. The terminating condition *term* determines whether the loop should stop or continue iterating, even if no better plan has been found. This terminating condition could depend on the total number of iterations, the time elapsed from the start of the search, or some other context-dependent condition. Once a better plan is found, this algorithm can be executed again using the new plan as the input seed plan π^*, in order to improve the plan quality further.

The initial seed plan can be an invalid one. This algorithm will compute the number of violations in the seed plan and try to find valid plans that do not have any violations. This makes it also suitable for plan repair, since it will exploit the plan structure of the initial plan to try to find a similar one that satisfies all the required conditions.

6. Preliminary Experiments

The proposed algorithm has been implemented within a temporal plan solver capable of parsing PDDL 2.2 domain and problem files, together with a seed plan for the problem. This solver performs a local search and outputs a new improved plan. If no seed plan is provided, it will try to find a feasible solution to the problem, which will then be used as the seed plan for subsequent iterations.

Some preliminary experiments have been performed with two temporal domains; the Transport domain, from the IPC 2008 competition, and the Aggregator domain, specifically designed to find solutions for managing demand-side electrical loads. Since the original version of the Transport domain was designed to minimise time, it has been slightly modified to also keep track of the total fuel used. The problem instances were also modified to make alternative cheaper or more expensive solutions also possible.

The aggregator domain represents a demand-side electricity aggregator that needs to schedule flexible load (such as dish-washing or EV charging), and also use storage devices to shift load to more preferable times of the day. Both the forecasted inflexible load and wholesale electricity prices fluctuate throughout the day and the goal is to find the best combination of actions that minimises the cost of inflexible and flexible load. The cost of the plan is calculated by adding the inflexible and flexible load components and multiplying the result with the energy cost at that time. This domain is particularly challenging for existent planners due to the fact that this cost depends on various variables that are changing over time rather than accumulating a monotonically increasing cost with each action.

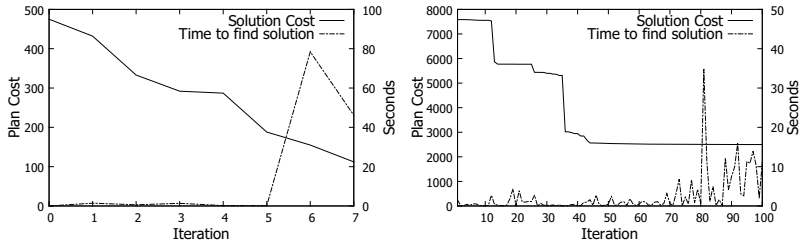
Table 1 shows an example of how an initial seed plan generated by the planner POPF [10] for the Aggregator domain was improved to minimise the costs by moving activities to cheaper periods and making use of batteries. Each line indicates a separate durative action in the plan. The number before the action indicates its start time, while the number in square brackets indicates its duration.

Table 1. Initial seed plan and improved plan for the Aggregator domain.

Seed Plan	`0.000: (start-metering) [1440.000]`
	`0.001: (perform wash-dishes-h3 wash-dishes-normal) [110.000]`
	`0.001: (perform wash-dishes-h2 wash-dishes-fast) [80.000]`
	`0.001: (perform wash-dishes-h1 wash-dishes-fast) [90.000]`

Improved Plan	`0.0: (start-metering) [1440.0]`
	`251.222: (charge battery-s1 charge-normal) [166.667]`
	`296.667: (perform wash-dishes-h2 wash-dishes-fast) [80.0]`
	`298.333: (charge battery-s2 charge-normal) [66.667]`
	`308.944: (perform wash-dishes-h3 wash-dishes-normal) [110.0]`
	`315.0: (perform wash-dishes-h1 wash-dishes-fast) [90.0]`
	`626.667: (discharge battery-s1 discharge-fast) [83.333]`
	`630.0: (discharge battery-s2 discharge-fast) [44.444]`

Figure 1a shows the plan improvements when running the proposed algorithm on a problem instance of the Transport domain with 5 cities, 2 trucks and 2 packages. Figure 1b shows the plan improvements for the Aggregator domain with 10 household tasks, 10 electricity storage units and 6 tariff switches over 24 hours. One should keep in mind that the potential plan improvement is naturally problem dependent and Figure 1 only demonstrates that for temporal numeric planning problems that have a broad solution space this approach is capable of improving plan quality.



(a) Transport domain (b) Aggregator domain

Figure 1. Improvements in plan costs with respect to the number of iterations.

7. Conclusions and Future Work

We have presented an algorithm that improves the quality of plans for temporal planning problems with numeric properties. While current state-of-the-art temporal planners are very capable of finding feasible plans, the proposed algorithm is able to exploit the structure of such plans to find better solutions. Preliminary experiments have been performed using two temporal domains and it has been demonstrated that this algorithm can improve plan quality, albeit more effective in problems with a broad solution space.

Future work includes incorporating techniques that shorten the makespan of a plan when it does not have any impact on the plan cost and inter-period optimisation to find the best schedule of a sequence of actions. Support for additional PDDL 2.1 constructs such as continuous effects and duration inequalities is also being investigated, together with further experiments with other domains using temporal and numeric characteristics.

References

[1] M. Fox and D. Long, "PDDL2.1: An extension to PDDL for expressing temporal planning domains," *Journal of Artificial Intelligence Research*, vol. 20, pp. 61–124, 2003.

[2] S. Edelkamp and J. Hoffmann, "PDDL2.2: The language for the classical part of the 4th international planning competition," Tech. Rep. 195, 2004.

[3] J. Bajada, M. Fox, and D. Long, "Challenges in Temporal Planning for Aggregate Load Management of Household Electricity Demand," in *31st Workshop of the UK Planning & Scheduling Special Interest Group (PlanSIG)*, 2014.

[4] A. Gerevini, A. Saetti, and I. Serina, "Planning Through Stochastic Local Search and Temporal Action Graphs in LPG.," *Journal of Artificial Intelligence Research*, vol. 20, pp. 239–290, 2003.

[5] A. E. Gerevini, A. Saetti, and I. Serina, "Temporal Planning with Problems Requiring Concurrency through Action Graphs and Local Search," in *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS 2010)*, no. Icaps, pp. 226–229, 2010.

[6] H. Nakhost and M. Martin, "Action Elimination and Plan Neighborhood Graph Search : Two Algorithms for Plan Improvement," in *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS 2010)*, pp. 121–128, 2010.

[7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing.," *Science (New York, N.Y.)*, vol. 220, pp. 671–80, May 1983.

[8] A. Coles, M. Fox, D. Long, and A. Smith, "Planning with Problems Requiring Temporal Coordination," in *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI-08)*, pp. 892–897, 2008.

[9] W. Cushing, S. Kambhampati, Mausam, and D. S. Weld, "When is Temporal Planning Really Temporal?," in *20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2007.

[10] A. Coles, M. Fox, and D. Long, "POPF2: a Forward-Chaining Partial Order Planner," *The 2011 International Planning Competition*, pp. 65–70, 2011.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-51

HiPOP: Hierarchical Partial-Order Planning

Patrick BECHON , Magali BARBIER , Guillaume INFANTES , Charles LESIRE and Vincent VIDAL
Onera — The French Aerospace Lab; F-31055, Toulouse, France;
name.surname@onera.fr

Abstract. This paper describes a new planner, HiPOP (Hierarchical Partial-Order Planner), which is domain-configurable and uses POP techniques to create hierarchical time-flexible plans. HiPOP takes as inputs a description of a domain, a problem, and some optional user-defined search-control knowledge. This additional knowledge takes the form of a set of abstract actions with optional methods to achieve them. HiPOP uses this knowledge to enrich the output by providing a hierarchical time-flexible partial-order plan that follows the given methods. We show in this paper how to use this additional knowledge in a POP algorithm and provide results on a domain with a strong hierarchy of actions. We compare our approach with other temporal planners on this domain.

1. Introduction

The main focus of our approach is to deal with multi-agent missions where several teams of robots must collaborate and schedule concurrent tasks to achieve a common goal. In some cases, for instance for sea rescue [1], it is also compulsory to follow known patterns. This is especially appropriate when the system has to interact with humans trained to follow certain procedures. In this paper, we are only concerned with the initial plan production but our approach is designed to be easily used in cases where there is a need to execute and repair the plan.

In order to achieve those goals, we designed a planner that will output a time-flexible plan with hierarchical actions. A time-flexible plan will be easier to execute and to repair since small delays in actions can be dealt with without replanning everything. The hierarchical structure of the actions will allow the user to provide additional knowledge to the planner to improve the planning time and to impose additional constraints such as following some procedures.

We can use the hierarchical actions to plan on a higher level, for instance with teams or robots. And then use this plan at the team level to instantiate it into a plan for every robot. This leads to solutions where robots on the same team share the same high-level goal at any time (which is not expressible in term of low-level action) and they move together as much as possible. For a given high-level goal we can also describe how to achieve it with elementary actions. This also allows

the output of a higher-level solution to a human operator while each robot has the full plan with all elementary actions.

These reasons lead us to investigate in this first step both partial-order planning and hierarchical task network planning, aiming at obtaining a new algorithm that will mix the best of both worlds. The unification of those two approaches has already been discussed under the term *hybrid planning*.

The goal of HiPOP is to output a partial-order plan, with time flexibility and time concurrent actions, including a hierarchy among actions. The additional user-provided knowledge is also used to improve the planner performance. This knowledge being optional, HiPOP will resort to a POP algorithm if none is provided.

2. Background and related work

POP (Partial-Order Planning) is an algorithm already used by several planners such as VHPOP [20] and CPT [19]. One of the main drawbacks of POP compared to other commonly used algorithms is that it is usually slower [20]. But partial-order plans are more convenient to use in plan reparation [10], plan merging [8] and plan adaptation [11]. They can also output time-flexible plans.

On the other hand, introducing higher-level (abstract) actions like in Hierarchical Task Network (HTN) planning has been shown to improve planning time in some cases [13] and to help during plan reparation [6]. It is also more expressive than first principle planners [5]. Since the first HTN planners were state-based, they usually could not deal with time constraints and with concurrent actions. Some formalisms have been proposed to deal with those conditions, such as [2,7,13].

The idea of HiPOP to mix those two approaches was inspired by DPOCL [21], which introduced decomposition of actions in the context of POP. The same idea was already studied under the name *hybrid planning*, including the extension of hierarchical planning in UCP [9] and the PANDA system [16]. PANDA is a formal framework meant to compare different algorithms and heuristics, but not meant to be used in a real world setting.

This work can also be related to several ideas used in other situations. Adding preconditions and effects to higher-level actions to better control search is explored in Angelic planning [12]. GoDeL [17] uses optional user-defined knowledge to guide a landmark-based planner. Mixing HTN and POP in a domain-specific planner with a strict separation between several hierarchical levels is described in [3]. SIADEX [2] adds temporal reasoning into a HTN planner, keeping a forward-chaining algorithm.

HiPOP uses its POP component to output a time-flexible plan with concurrent actions, unlike state-based planners. It is able to plan with several levels of abstraction concurrently. And unlike other work on *hybrid planning*, HiPOP is implemented and compared to other temporal planners.

As HiPOP is an extension to classical POP, we first explain POP algorithm before introducing our additions. We then describe the search-control heuristics used, before explaining our experimental setup and showing some results.

3. Classical POP algorithm

We describe here a plain fully instantiated Partial-Order Planning algorithm (also called Partial-Order Causal Link). We selected a literal-based description of the world where a state is represented as a set of positive literals. The negation of literal l is noted $\neg l$. A domain is defined as a set of actions. The resulting state of the application of an action is obtained from a starting state by removing the set of its negative effects and adding the positive ones.

Definition 1 (action). *An action is a tuple* $(\mathcal{P}, \mathcal{E}, dur)$ *where* $\mathcal{P}$ *is a set of literals representing the preconditions,* $\mathcal{E}$ *is the set of literals representing the effects and* dur *is the duration of the action.*

Definition 2 (step). *A step* s *is a tuple* $s = (a, t_s)$ *where* $a = (\mathcal{P}, \mathcal{E}, dur)$ *is an action and* t_s *is an index of a time point in an STN. This timepoint represents the start time of* s*. We denote* $act(s) = a$*,* $t_{start}(s) = t_s$*,* $t_{end}(s) = t_s + dur$*,* $\mathcal{E}(s) = \mathcal{E}$*,* $\mathcal{P}(s) = \mathcal{P}$ *and* $dur(s) = dur$*.*

$t_{end}(s)$ represents the end time of the step s. It is used as a convenience notation to represent time constraints that only deal with the start time of each step. A Simple Temporal Network (STN) [4] is used to check schedulability over the time point indexes. If the set of constraints allows at least one solution, the STN (or equivalently the set of constraints) is said to be *consistent*.

Let s_i, s_j be steps. $s_i \prec s_j$ is a shorthand for $t_{end}(s_i) \leq t_{start}(s_j)$. We will use the classical definitions of causal links (noted $(s_i \xrightarrow{l} s_j)$ where l is a literal), open links (noted $(\xrightarrow{l} s_i)$) and threats (noted as a tuple $(s_k, s_i \xrightarrow{l} s_j)$ where s_k is the threatening step and $s_i \xrightarrow{l} s_j$ is the threatened causal link).

Definition 3 (flaw). *A flaw is either an open link or a threat.*

Definition 4 (partial plan). *A partial plan* P *is a tuple* $(\mathcal{S}, \mathcal{TC}, \mathcal{CL}, \mathcal{F})$ *where* $\mathcal{S}$ *is a set of steps,* $\mathcal{TC}$ *is a set of (simple temporal) constraints over the time points of* $\mathcal{S}$*,* $\mathcal{CL}$ *is a set of causal links,* $\mathcal{F}$ *is a set of flaws. We denote* $\mathcal{S}(P) = \mathcal{S}$ *and* $\mathcal{F}(P) = \mathcal{F}$*.*

A POP algorithm will explore the space of partial plans to find a complete plan. P is said to be *consistent* if $\mathcal{TC}(P)$ is consistent. P is said to be *complete* if $\mathcal{F}(P) = \emptyset$ and P is consistent.

Definition 5 (planning problem). *A problem instance is a tuple* $(\mathcal{A}, I, G)$ *where* $\mathcal{A}$ *is the set of available actions,* I *is a set of literals representing the initial state,* G *is a set of literals representing the goal.*

Algorithm 1 shows the pseudocode for a POP algorithm solving a planning problem $Prb = \{\mathcal{A}, I, G\}$. It works by keeping a set of all the generated but not yet visited plans: Π.

The initial plan is built by the procedure *InitialPartialPlan*. It creates a plan with two dummy steps, corresponding to actions a_s and a_e. a_s is the dummy start action with $\mathcal{P}(a_s) = \emptyset$, $\mathcal{E}(a_s) = I$ and a_e is the dummy end action with

Algorithm 1: Basic POP algorithm

```
1  Π = {InitialPartialPlan(I, G)} ;
2  while Π ≠ ∅ do
3  |   P = PopBestPlan(Π) ;
4  |   if F(P) = ∅ then
5  |   |   return P ;
6  |   end
7  |   f = PopBestFlaw(F(P)) ;
8  |   Π = Π ∪ Resolvers(A, P, f) ;
9  end
10 return ∅
```

$\mathcal{P}(a_e) = G$, $\mathcal{E}(a_e) = \emptyset$. All other steps must appears after the dummy start step and the dummy end step.

The *PopBestPlan* procedure removes the best partial plan from Π according to a heuristic and returns it. *PopBestFlaw* does the same with the set of flaws. At each iteration the algorithm selects the next plan to expand (line 3). Then a flaw is chosen (line 7) and the successors of P are generated and added to Π (line 8). The *Resolvers* $(\mathcal{A}, P, f)$ procedure returns a set of partial plans, each consistent and solving f in P in a different way accordingly to the type of the flaw. The algorithm stops when a complete plan is found (line 5) or when Π is empty (line 10).

To remove an open link to s, the *Resolvers* procedure has to add a causal link from s_i. s_i can be an existing step in P or a newly introduced step built from an action of $\mathcal{A}$. When adding a new causal link new threats can appear. When adding a new step all of its preconditions must be added as open links.

To remove a threat $(s_k, s_i \xrightarrow{l} s_j)$, there are only two ways: either the constraint $s_k \prec s_i$ (*demotion*) or $s_j \prec s_k$ (*promotion*) has to be added to the STN.

4. Adding abstract actions to classical POP

The goal of HiPOP is to use higher-level actions during search. The planner should be able to use abstract steps as elementary steps and to refine them when needed into a set of steps, causal links and temporal constraints.

Definition 6 (abstract action)**.** *An abstract action, also called higher-level action, is a tuple* $(\mathcal{P}, \mathcal{E}, dur, \mathcal{M}, \mathcal{C})$*:*

- *the first three elements* $(\mathcal{P}, \mathcal{E}, dur)$ *are the same as in an elementary action (Definition 1),*
- $\mathcal{M}$ *is a set of partial plans (called methods), used to instantiate the action,*
- $\mathcal{C}$ *is a set of conflicts (see Definition 8 below).*

A step with an abstract action is called an abstract step. The method of an abstract action is a partial plan in itself. During search, abstract actions can be

used as any other actions. But a new type of flaw is introduced: the abstract flaw. It represents the fact that there is an abstract step in the plan.

Definition 7 (flaw in abstract POP)**.** *(Replaces Definition 3) A flaw is an open link, a threat or the presence of an abstract step in P (abstract flaw).*

The only way to solve this new flaw is to pick one of its methods and to introduce the partial plan representing it in the current plan. It means adding all the steps, causal links, temporal constraints and flaws of the method to the plan, along with the dummy actions (see below). When adding the steps, a new time point is created for each one. After that, the abstract step has no effect on search (it cannot be used as the origin of a new causal link), but the newly created (non-dummy) steps can be used normally. The hierarchy of steps (which steps were introduced as children of which other steps) is also kept and returned at the end of the algorithm. Algorithm 1 is still valid, but *Resolvers* have to be adapted to deal with this new type of flaw.

Dummy actions (and their corresponding steps) are introduced in every partial plan. They are slightly different in case of methods associated to the abstract action a_i: the dummy actions a_i^s and a_i^e are such that $\mathcal{P}(a_i^s) = \mathcal{E}(a_i^s) = \mathcal{P}(a_i)$ and $\mathcal{P}(a_i^e) = \mathcal{E}(a_i^e) = \mathcal{E}(a_i)$; their duration is null. The difference with the dummy actions introduced in the initial plan is that the initial action has now a set of preconditions and the end dummy action has a set of effects.

They are introduced to deal with the following case. Assume that the open link $\xrightarrow{l} s_i$ exists when s_i is instantiated, where s_i is a step using a_i. Assume also that l is used by several steps in s_i. Then each step of s_i can be linked to the dummy initial step s_i^s. This open link is then the only one needed to guarantee that all the requirements of the child actions are met. Without dummy actions, we would have an open link for each step using l, increasing the number of plans to explore to solve them. This also guarantees than the same provider of l will be used for all steps in s_i.

Allowed actions. Using HiPOP as presented above leads to very poor performance and the output does not always take advantage of the hierarchical description of actions. This is because we only increased the branching factor, but even if the abstract actions are efficient the algorithm will explore in parallel plans with and without abstract actions. Those branches will produce the same elementary plan but will be explored concurrently.

To solve this issue we used an idea from HTN planning: the user provides a set of highest-level actions, the only actions that the planner can use to add a step. All the other actions are only used when refining an existing step. So the only change is in the *Resolvers* procedure, when adding a new step the algorithm can only select an explicitly allowed action. Planners like TALPlan and TLPlan also use the additional knowledge to prune the search tree, and this is similar to the search of plan respecting the *user-intent* presented in [9].

Threats of abstract actions. Some issues arise that can over-constrain the problem if threat solving procedures are not adapted to deal with abstract steps. Let us consider a threat $(s_k, s_i \xrightarrow{l} s_j)$. If s_k is abstract and encompasses several steps, it might be enough to promote only the last one and not all of them. If s_i is abstract and encompasses several steps, it might be enough to demote s_k before

the last step and not before all of them. If s_j is abstract and encompasses several steps, it might be enough to promote s_k after the first step and not after all of them. To deal with this problem, we use a new kind of promotion. The idea is to add only the mandatory constraint before the refinement even if it means that another constraint will need to be added after.

The constraint for demotion is $t^k_{dem} < t^i_{dem}$ and the constraint for promotion is $t^j_{pro} < t^k_{pro}$ where each variable, defined below, depends on the fact that the respective steps involved are elementary or abstract. If any step is abstract, the constraint will be loosened compared to the one enforced by the previous definition. If all the steps are elementary, the definitions are identical to the previous algorithm. It is generally not enough to ensure that the plan will be consistent. The algorithm will have to wait until the refinement to compute more precise threats.

- $t^i_{dem} \leftarrow t_{start}(s_i)$ **if** s_i is elementary **else** $t_{end}(s_i)$
- $t^k_{dem} \leftarrow t_{end}(s_k)$ **if** s_k is elementary **else** $t_{start}(s_k)$
- $t^j_{pro} \leftarrow t_{end}(s_j)$ **if** s_j is elementary **else** $t_{start}(s_j)$
- $t^k_{pro} \leftarrow t_{start}(s_k)$ **if** s_k is elementary **else** $t_{end}(s_k)$

If s_i is abstract we want to avoid scheduling s_k before the whole abstract step, so we restrict the demotion constraint to consider $t_{end}(s_i)$ instead of $t_{start}(s_i)$. If s_j is abstract we want to avoid scheduling s_k after the whole abstract step, so we restrict the promotion constraint to consider $t_{start}(s_j)$ instead of $t_{end}(s_j)$.If s_k is abstract we want to allow the promotion of only the last step of s_k (line 4) after s_j and the demotion of only the first step of s_k before s_i.

Another problem arises from the fact that a literal can be destroyed and recreated inside an abstract action. For instance in the *survivors* domain, one can create an abstract action where each team has a hospital as a homebase. Each step of this abstract action will move the team from its homebase to a survivor and back to the hospital. So the preconditions and the effects will have *(at ?team ?homebase)* but no causal link on the position of the team can be valid through the abstract step. It is inefficient to wait until the refinement of the abstract step to detect it.

To solve this issue, we introduced the notion of *resource conflicts*. Resource conflicts are provided in the description of abstract actions. They serve as a way to find which causal link does an abstract step threaten, independently of its effects. This also allows to detect two abstract steps that cannot appear concurrently.

Definition 8 (resource conflict)**.** *A resource conflict is a (hand-given) set of literals for an abstract action that may threaten causal links (but are not necessary a negative effect of the abstract action).*

Definition 9 (abstract resource conflict)**.** *Two abstract steps are in abstract resource conflict if the intersection of their resource conflicts is not empty.*

Abstract resource conflicts are considered as a new type of flaw. They can be solved in the same way than threats: removing a resource conflict between a_i and a_j if a_i and a_j are abstract is done by adding either $a_i \prec a_j$ or $a_j \prec a_i$ to the set of temporal constraints.

Soundness. The soundness proof of HiPOP follows the same pattern than the soundness proof of POP. If the algorithm returns a plan, one can remove all the abstract steps from this plan and only consider the elementary steps. The chain of causal links and the absence of threats are a guarantee that the plan is executable and accomplishes the goal.
Completeness. It highly depends on the available description and hypothesis. For instance literals can be masked by the hierarchical description if we assume that the elementary actions are forbidden. The proof is quick and easy if we allow the algorithm to plan with abstract or elementary actions without restriction, but this does not represent the actual use of the algorithm. The search is complete among the space of all plans that can be represented using only allowed action at the higher level, ie. among the plans respecting the *user intend*.

5. Search control

Algorithm 1 uses two heuristics to control search. On line 3 a first one is used to choose the next plan that will be expanded, called the *plan heuristic*. On line 7 another one is used to choose the next flaw that will be solved in a given plan, called the *flaw heuristic*. Those two heuristics are highly critical for the efficiency of HiPOP.
Plan heuristics. HiPOP uses the A^* algorithm to sort the set Π of all plans generated but not yet explored. They are stored in increasing order of $f(P) = g(P) + h(P)$ where $g(P)$ is the "distance" from the start point and $h(P)$ is a heuristic estimation of the cost to reach a complete plan from P. In HiPOP $g(P)$ is an estimation of the number of elementary (non-dummy) steps in P. It is an estimation because we cannot be sure of the number of steps in an abstract step if several methods are available to refine it. In this case, the minimum is taken. It can thus be defined recursively as:

$$g(P) = \sum_{s \in \mathcal{S}(P)} \begin{cases} 1 & \text{if } s \text{ is elementary} \\ \min\limits_{m \in \mathcal{M}(act(s))} g(m) & \text{if } s \text{ is abstract} \end{cases} \quad (1)$$

The computation of $h(P)$ uses the h_{add} heuristic as described by VHPOP [20]. Due to the lack of space we cannot describe it here, but we used the sum of the *cost* of each open link to sort plan, breaking ties using the *effort*.

The h_{add} heuristic does not take into account action reuse, and VHPOP proposes a modification of h_{add} to partially take care of it, called *reuse*.
Flaw heuristics. Previous work on VHPOP [20] as well as our initial results show that solving threat first is usually a good heuristic, especially when choosing first the ones with the fewest available resolvers.

To choose between open link flaws and abstract flaws, it is possible to always pick first abstract flaws or to mix their resolution with open link flaw or to pick them last.

Picking them first means that there is almost no planning done with abstract steps, but rather than they are used as a template of what steps should be created

together. Our tests showed that this leads to poor results since we do not have the benefits of planning with abstract steps but have to refine a lot of them.

Mixing the resolution of open link and abstract flaws is not yet studied in HiPOP due to the lack of heuristics that would allow to compare them. Instead, we focused on heuristics where the abstract flaws are solved at the end to plan as long as possible with abstract steps. This means that the planner will first compute an abstract plan, a plan whose only flaws are abstract, before refining them. This allows the planner to deal with smaller plans during the search, thus reducing the planning time until an abstract plan is found. If the abstract description ensures that any valid abstract plan can be refined into a complete plan without much backtrack, refining this abstract plan can be done quickly. And it also allows to separate the search on each instantiation of an abstract action.

While refining abstract steps, it is possible that some literals get "hidden" by the description: they are created by the elementary steps but are not in the abstract description (for instance they can depend on the choice of which method to use for this step). If those literals are needed to finish the plan, this can be an issue for the planner. To avoid this, the abstract flaws can be refined in their chronological partial order. If no literal is hidden and if the description allows a backtrack-free refinement, then the solving order does not matter.

To sort the open links, we used the same heuristics as the one described by VHPOP namely MW-Loc. It ranks open link according to the *effort* of their literals for the most recently added step. This is a compromise between staying focused on the current sub-goal and solving the harder literals first.

All the following results uses MW-Loc. It means that plans are sorted with $A*$ using the remaining *cost* as a heuristic. If the *costs* are equal, the remaining *effort* is used as a tie-breaker. The flaws are selected first on their type: *threats* first, then *open links* then *abstract flaw*. *Threats* are sorted in LIFO order. The first *open link* to be chosen is the one coming from the most recently added step, with the *effort* of this open link used as a tie-breaker. *Abstract flaws* are solved in chronological order (this is only a partial order, if two steps can be scheduled together, the flaws are solved in LIFO order).

6. Experimentation and results

We implemented HiPOP in C++, using the IPPC algorithm [14] to incrementally solve STNs. The domains and problems are modeled in PDDL and the abstract actions definition in a PDDL-like language. We created a random generator of problems for the *survivors* domain and a description of hierarchical actions. It is possible to vary the number of teams, of hospitals, of survivors and the size of zones. The position of hospitals and survivors are randomly chosen.

The goal of the hierarchical actions is to allow HiPOP to plan first by reasoning with teams (instead of individual robots) and zones (instead of individual locations). Once a plan with only abstract flaws is encountered, every abstract action will be instantiated into a set of elementary actions. Those elementary actions will be concerned with individual robots.

The only actions the planner is allowed to insert to solve an open link are elementary moves of a robot or a team, a hierarchical action to use a team to

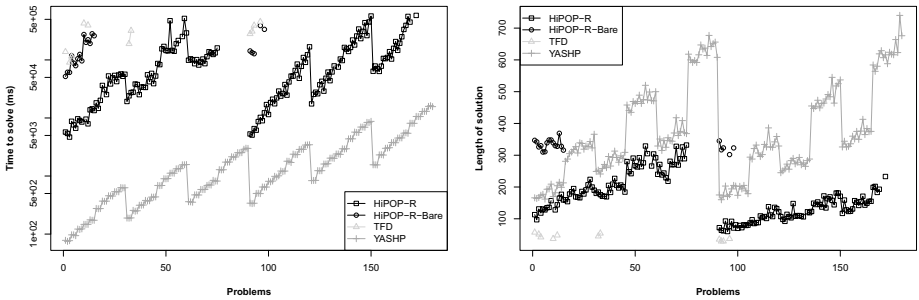


Figure 1. Time to solve (left) and makespan (right) on the *survivors* domain. HiPOP-R uses the *reuse* heuristics. HiPOP-R-Bare uses the same heuristics but without abstract actions. We also included the results of TFD (Temporal Fast Downward) and YASHP

explore a zone and a hierarchical action to use a team to bring a survivor back to a hospital. For each exploration action, there is one pre-computed patrol for a team to explore a zone that is not necessarily optimal. This means that the planner must still make sure that all actions are correctly chained up but does not need to solve a multi-vehicle travelling salesman problem on the whole zone. For each action bringing back a survivor, only a skeleton is given and the planner has to add some motion actions to create a valid plan.

We ran four different planners on a set of 180 randomly generated problems in the *survivors* domain. We used two versions of HiPOP, with the *reuse* heuristics : HiPOP-R and HiPOP-R-Bare. The latter does not use user-defined knowledge, so it is a classical POP algorithm. We also used two other temporal planners used in the temporal track of the IPC: Temporal Fast Downward [15] and YASHP [18]. The experiments were all run on an Intel X5670 processor running at 2.93Ghz with 24GB of RAM and a timeout of 10 min. The results are shown on Figure 1.

First we can see that adding the additional knowledge to HiPOP helps to reduce the planning time (by a factor of 10 on the first problems) and to improve the quality of the output plan. This mostly comes from the fact that HiPOP does not have to solve a generic travelling salesman problem but can use the precomputed patrol to solve it more quickly (even if it means that it cannot find the optimal solution, it will only find solutions that use those patrols).

Comparing it to other planners we can see that YASHP can very quickly find a solution that is almost always of poorer quality than that of HiPOP. On the opposite TFD is always slower than HiPOP but finds good solutions. The solutions found by TFD do not respect the abstract decomposition so they are not available to HiPOP.

7. Conclusion

We have shown how to add additional user knowledge into a POP planner and how this knowledge can be used by the planner to improve its performance. We presented HiPOP, a planning algorithm that uses this knowledge to output hierarchical, temporally flexible plans. The output plan also provides the information

about all the hierarchical actions that were used to generate the plan, so for each elementary action we can know the hierarchical action it belongs to.

For future work, we will further improve the performance of HiPOP. Several heuristics could be useful as shown by other research on POP algorithm, such as sorting the flaws by their number of resolvers. Another direction of study will also be to use the enriched plans to repair or merge them. It can also be used to better control its execution, in addition to the flexibility added by the least-commitment principle for temporal constraints used by POP.

References

[1] *International aeronautical and maritime search and rescue manual*, volume 3. IMO Publishing, 1998.

[2] L Castillo, J Fdez-Olivares, O Garcia-Pérez, and F Palao. Efficiently handling temporal knowledge in an HTN planner. In *ICAPS*, 2006.

[3] L Castillo, J Fdez-Olivares, and A Gonzalez. Integrating hierarchical and conditional planning techniques into a software design process for automated manufacturing. *Workshop on Planning under Uncertainty and Incomplete Information, ICAPS*, 2003.

[4] R Dechter, I Meiri, and J Pearl. Temporal constraint networks. *Artificial intelligence*, 49(1):61–95, 1991.

[5] Kutluhan Erol, James Hendler, and Dana S Nau. Htn planning: Complexity and expressivity. In *AAAI*, volume 94, pages 1123–1128, 1994.

[6] T Gateau, C Lesire, and M Barbier. HiDDeN: Cooperative Plan Execution and Repair for Heterogeneous Robots in Dynamic Environments. In *IROS*, Tokyo, Japan, 2013.

[7] RP Goldman. Durative Planning in HTNs. *ICAPS*, 2006.

[8] M A Hashmi and A El Fallah Seghrouchni. Merging of Temporal Plans Supported by Plan Repairing. *ICTAI*, 2010.

[9] S Kambhampati, A Mali, and B Srivastava. Hybrid planning for partially hierarchical domains. *AAAI/IAAI*, pages 882—-888, 1998.

[10] Roman Van Der Krogt and Mathijs De Weerdt. Plan Repair as an Extension of Planning. *ICAPS*, pages 161–170, 2005.

[11] S M Lee-Urban. *Hierarchical Planning Knowledge for Refining Partial-Order Plans*. PhD thesis, Lehigh University, 2012.

[12] B Marthi, S J Russell, and J Wolfe. Angelic semantics for high-level actions. In *ICAPS*, 2007.

[13] D Nau, T C Au, O Ilghami, U Kuter, J W Murdock, D Wu, and F Yaman. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, 20(1):379–404, 2003.

[14] L Planken, M de Weerdt, and N Yorke-Smith. Incrementally solving STNs by enforcing partial path consistency. *ICAPS*, 2010.

[15] G Röger, P Eyerich, and R Mattmüller. Tfd: A numeric temporal extension to fast downward. *6th IPC planners descriptions*, 2008.

[16] B Schattenberg. *Hybrid Planning And Scheduling*. PhD thesis, Ulm University, Institute of Artificial Intelligence, 2009.

[17] V Shivashankar, R Alford, U Kuter, and D Nau. The GoDeL Planning System: A More Perfect Union of Domain-Independent and Hierarchical Planning. *IJCAI*, 2013.

[18] V Vidal. YAHSP2: Keep it simple, stupid. *IPC*, pages 83–90, 2011.

[19] V Vidal and H Geffner. Branching and pruning: An optimal temporal POCL planner based on constraint programming. *Artificial Intelligence*, 170(3):298–335, 2006.

[20] HLS Younes and RG Simmons. VHPOP: Versatile heuristic partial order planner. *Journal on Artificial Intelligence Research (JAIR)*, 20:405–430, 2003.

[21] RM Young, ME Pollack, and JD Moore. Decomposition and causality in partial-order planning. *International Conference on AI and Planning Systems (AIPS)*, 1994.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-61

Value Iteration for Relational MDPs in Rewriting Logic

Lenz BELZNER [a,1]
[a] *LMU Munich, PST Chair*

Abstract. Relational approaches to represent and solve MDPs exploit structure that is inherent to modelled domains in order to avoid or at least reduce the impact of the *curse of dimensionality* that propositional representations suffer from. By explicitly reasoning about relational structure, policies that lead to specific goals can be derived on a very general, abstract level; thus, these policies are valid for numerous domain instantiations, regardless of the particular objects participating in it. This paper describes the encoding of relational MDPs as rewrite theories, allowing for highly domain-specific domain encoding and abstraction. Narrowing is employed to solve these relational MDPs symbolically. Resulting symbolic value functions are simplified by Ax-matching abstract state terms. It is shown that relational state representations significantly reduce the size of state space and value function when compared to propositional representations.

Keywords. RMDP, Symbolic Value Iteration, Rewriting Logic, Narrowing

1. Introduction

The framework of *Markov decision processes* (MDPs) allows to model domains with non-deterministic action outcomes and arbitrary reward functions, thus serving well for modelling problems of *sequential decision making under uncertainty* [15]. Various exact and approximate techniques to solve MDPs exist, such as *value iteration*, *policy iteration* and *modified policy iteration* [18]. Given a reward specification (i.e. system goals), a solution of a MDP can be computed that is either a value function mapping states to their corresponding expected values (according to the reward function of the MDP) or a policy mapping states to actions that are maximizing the expected reward as the policy is executed. Algorithms for solving MDPs suffer from the *curse of dimensionality* [1], rendering them infeasible for large-scale domains. To overcome this problem, effort has been made to exploit inherent structure of domains by employing factored or relational, first-order representations of states and actions instead of propositional ones, allowing to represent structured problem domains more concisely. Thus, computation becomes feasible also for larger domains, but the additional complexity that arises from structured domain representations has to accounted for when solving the according MDP [3,12,16].

Rewriting logic is a formal logical framework that lends itself naturally to modelling non-deterministic and concurrent domains on a symbolic level [14,5]. It provides

[1] This work has been partially funded by the EU project ASCENS, 257414.

support for formally specifying structured domains through modularization and object-orientation, as well as explicitly ordered sorts and polymorphism. Previous work of the author introduced the specification of relational MDPs (RMDPs) in rewriting logic [2]. This paper shows how matching [9] and narrowing [10,4] can be employed to solve them using first-order abstraction and avoiding propositionalization. A model-based dynamic programming algorithm for relational MDPs is introduced that employs first-order reasoning and computes exact solutions with the following properties:

- It operates on order-sorted state representations, allowing for polymorphism.
- User-definable constraints assure regression of valid states.
- Only goal-relevant abstract states are regressively constructed and evaluated.
- Reasoning is performed on the first-order level wherever possible.
- The state-action space is factored, leading to concise results.
- Partial goal specifications are supported.

Using rewriting logic to encode domains as RMDPs allows to incorporate features like user-definable term syntax and equivalence classes, sort-ordering, polymorphism and object-orientated representation [6]. This enables domain encoding with only a small representational difference to human expert knowledge. E.g., as many modern software systems are modelled according to the OO paradigm, this way to specify system autonomicity may help to brigde the gap between software engineering and AI techniques.

The paper is outlined as follows: Section 2 discusses in more detail value iteration to solve MDPs exactly as well as the rewriting logic framework and the concepts of matching and narrowing. Section 3 describes how RMDPs can be encoded in terms of a rewrite theory and how rewriting logic concepts can be used to solve these relational MDPs symbolically. Section 4 discusses experimental results and a visualization approach. Finally, section 5 compares the approach to related work, summarizes the results described in this paper and hints at possibilities for further research.

2. Preliminaries

2.1. MDPs and Value Iteration

Definition 2.1. A *Markov decision process (MDP)* is a tuple (S,A,T,γ,R) with S a set of states, A a set of actions, $T : S \times A \times S \to \mathbb{R}$ a transition function, $\gamma \in [0;1]$ a discount factor and $R : S \to \mathbb{R}$ a reward function.[2]

A tuple as given in definiton 2.1 specifies the non-deterministic, discrete time dynamics of a domain in terms of a transition system. The transition function T encodes the probability that executing an action $a \in A$ in a particular state $s \in S$ will result in a state $s' \in S$; note that s and s' may be equal, indicating absence of an action effect. The discount factor γ reflects how much an agent prefers immediate over long-term rewards; the smaller γ is chosen, the more immediate rewards will impact behaviour of an agent acting according to the MDP. The reward function defines incentives that are given to the agent in particular states; in other words, it specifies which states are valuable to achieve.

[2] State-based rewards are used for the sake of simplicity. The approach described in this paper can be extended straightforwardly to allow for transition-based rewards as well.

Definition 2.2. A *value function* $V : S \to \mathbb{R}$ maps states to values. The value of a state $s \in S$ is the reward gained in s plus the expected discounted future reward when acting greedily w.r.t. V. A *policy* $\pi : S \to A$ maps states to actions. An agent *acting according to a policy* π executes action $\pi(s)$ when being in state s.

Solving a MDP means to compute either a value function V mapping states to expected values w.r.t. the given reward function, or to provide a policy π that maps states in S to actions in A that are going to maximize the expected reward of an agent acting according to π.

$$V_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A} \left(\sum_{s' \in S} T(s,a,s') V_i(s') \right) \tag{1}$$

Equation (1) shows the *value iteration* algorithm that computes V iteratively, which is guaranteed to converge to the optimal solution [1]. The general idea is that the value of a state is the sum of the reward this state will expose to the agent and the expected discounted future reward when moving on to the next state by executing an action that is assumed to be optimal w.r.t. the current value function V_i. V_0 can be arbitrarily initialized, a common approach is to set $V_0(s) = R(s)$. Iteration is performed until $|V_{i+1}(s) - V_i(s)| < \varepsilon(1-\gamma)/\gamma$ for each state $s \in S$ and a given error bound $\varepsilon \in \mathbb{R}$. This ensures that the maximum difference of V_{i+1} to the real value function V is smaller than ε for all states [18].

2.2. Rewriting Logic

Rewriting logic is suited towards the formal specification of non-deterministic, concurrent systems. System states are encoded in user-definable terms that are constructed from specified operations that can be enhanced with axioms like associativity, commutativity or idempotency. System dynamics are then described in terms of so-called rewrite rules, that allow to specify non-deterministic and concurrent behaviour. The core element to describe systems in rewriting logic are *rewrite theories*, i.e. tuples of the form $(\Sigma, E \cup A, R)$, where Σ contains sorts and operations that are used to construct state terms, E is a set of equations that define equivalence classes for these terms, A is a set of axioms like associativity, commutativity or idempotency specified for operations in Σ, and R is a set of rewrite rules that define the system dynamics. A rewrite theory represents a transition system, where states are terms in Σ, and rewrite rules in R define state transitions. A central concept in rewriting logic is *matching*, which is performed modulo axioms and with extensions (*Ax-matching*, denoted $:=_{Ax}$). Consider an infix operation $\circ$ being associative and commutative, i.e. rendering terms constructed by means of $\circ$ into a *multiset*. Then, for example, $a \circ b :=_{Ax} a \circ c \circ b$, as the latter has the same equivalence class as the term $a \circ b \circ c$ (due to commutativity) and $a \circ b$ is a subterm of $a \circ b \circ c$.

While state terms in Σ specify the static representation of a system, its dynamics are formalized in terms of *rewrite rules* of the form $label : t \to t'$ **if** $Conditions$ where t and t' are terms of the same kind and may contain free sorted variables. If t Ax-matches a given subject term, the subject term's matched portions are rewritten to t'. A rule's label may be omitted. Rewrite rules can optionally be conditional, in which case rewriting only is applied if all conditions evaluate to true (e.g. sort tests, boolean conditions or matching and rewriting conditions checking whether a term has a certain structure w.r.t. rewrite

theory). As a rule may match different portions of a subject term, this representation of system dynamics offers a natural way to model concurrency by rewriting a term on various positions simultaneously according to one or multiple rewrite rules. On the other hand, non-determinism is expressed if (partially or completely) overlapping portions of a subject term match one or more rewrite rules, i.e. if different applications of rewrite rules are possible without being applicable concurrently. In this case, a subject term evolves non-deterministically in any possible way. For example, consider the rewrite rules $(i) : a \to a'$, $(ii) : a \to a''$ and $(iii) : b \to b'$. Then, the term $a \circ b$ rewrites to $a' \circ b'$ by applying rules (i) and (iii), and also to the form $a'' \circ b'$ (using rules (ii) and (iii)).

While rewriting treats variables in a rewriting problem universally quantified, i.e. answering a problem of the form $\forall \vec{x} : t(\vec{x}) \to_? t'(\vec{x})$, a technique called *narrowing* deals with corresponding problems where variables are treated existentially, i.e. $\exists \vec{x} : t(\vec{x}) \to_? t'(\vec{x})$, representing *symbolic reachability* problems. To answer queries of this form, instead of matching rules and subject terms as in rewriting, they are *unified* in order to perform narrowing, meaning that variables in both terms may be instantiated to achieve syntactic term unification. I.e., when narrowing, rewrite rules are applied if (one or more subterms of) the subject term can be unified with a rule's lefthand side. Note that, when narrowing, righthand sides of rewrite rules may contain variables not specified in their lefthand side, allowing rewrite rules to introduce fresh variables. For an in-depth discussion of rewriting logic and the concepts of matching, rewriting and narrowing see for example [6].

3. Value Iteration for Relational MDPs in Rewriting Logic

3.1. Relational MDPs in Rewriting Logic

In order to perform symbolic value iteration, a relational MDP (S, A, T, γ, R) is encoded as a rewrite theory $(\Sigma, E \cup A, R)$. States are represented as associative-commutative terms with user-definable syntax parametrizable with first-order variables, thus allowing to specify relations between domain objects and to avoid propositional representation where possible. Axioms and equations of the rewrite theory define equivalence classes for state (and action) terms, which in turn are representation of non-ground, abstract states. Also, state terms can be constrained by equations. This allows for a concise representation of MDPs even when there is a large number of domain objects. Also, as will be shown in section 3.2, a RMDP specified as a rewrite theory can be solved completely symbolically, only grounding variables where this is relevant for goal reachability.

For example, in a fluent-based representation, dynamically changing relations of domain objects can be encoded by a corresponding sort FLUENT. Negation of fluents (i.e. the explicit absence of a particular state property) is represented by an operation $\neg : \text{FLUENT} \to \text{FLUENT}$. The state space is constructed in terms of a sort STATE (with FLUENT being a subsort of sort STATE) by an associative and commutative operation $\wedge : \text{STATE} \times \text{STATE} \to \text{STATE}$ that is representing logical conjunction. A constant *false* is defined for sort STATE to denote constraint violations, and $\neg F \wedge F = \mathit{false}$ for all $F \in \text{FLUENT}$. State terms that are syntactically constructed in this way may still contain semantic inconsistencies. Semantic constraints (e.g. state invariants) can be specified in terms of equations that reduce states that violate constraints to *false*. States that violate constraints (e.g. invariants) are reduced to *false*: $S \wedge \mathit{false} = \mathit{false}$ for all $S \in \text{STATE}$. The

set of equivalence classes of state terms can then be considered the set S of states of a relational MDP. Note that each equivalence class may render different instances of state terms equal according to their relational structure, thus providing a representation of abstract first-order states. Primitive actions executable by agents are encoded by a parametrizable sort ACTION. Equivalence classes on terms of sort ACTION then form the set of actions A of a relational MDP. As for state terms, free variables are allowed in action terms. The action space is thus raised to an abstract level that allows to exploit its relational structure, especially when taking into account relations between state and action space, e.g. if state and action terms share free variables.

Example 3.1. Consider a signature with sorts TRUCK, BOX and CITY and the sorts FLUENT, STATE and ACTION as above. One can then for example define a fluent $on : \text{BOX} \times \text{TRUCK} \rightarrow \text{FLUENT}$. Consider e.g. a polymorph fluent *in* defined similarly. Then $in(t,c) \wedge on(b,t) \wedge in(b',c)$ is a term of sort STATE in Σ.[3] An action representing a truck loading a box can be defined by an operation $load : \text{TRUCK} \times \text{BOX} \rightarrow \text{ACTION}$. Then, $load(t,b)$ is an ACTION-term in Σ. The constraint that a truck can only be in one city at a time can be specified by a conditional equation $in(T,C) \wedge in(T,C') \wedge S = false$ **if** $C \neq C'$. ▲

To encode the relation of states, actions (e.g. an optimal action in a particular state) and any corresponding values (e.g. a state's probability to be reached or its expected value), *SAV-tuples* (state-action-value tuples) of the form (s,a,v) are defined. When either a or v are omitted, the relation is valid for all actions or values, respectively. Note that SAV-tuples may relate state and action terms that share variables, consider for example a SAV-tuple $(s(\vec{x} \cup \vec{y}), a(\vec{y} \cup \vec{z}), v)$ where state and action share the variables $\vec{y}$. Thus, SAV-tuples allow to partition not only the state space, but the combined state-action space.

To model the transition function T of a MDP, a rewrite rule can be defined for any transition $T(s,a,s') = p$ to specify this transition in a rewrite theory[5]. The transitions in T for an action $a \in A$ are encoded as a disjunctive rewrite rule (with $\sum p_i = 1$; considering $\vee$ as disjunctive constructor for sets of SAV-tuples):

$$(s,a) \rightarrow (s'_1, p_1) \vee (s'_2, p_2) \vee \ldots \vee (s'_n, p_n) \ .$$

The state terms s and the s'_i can be considered as pre- and postconditions of action a. This representation of domain dynamics provides a solution to the *frame problem* [13], avoiding the necessity to specify all state properties unchanged by action execution.

Finally, a MDP's reward function R is represented in terms of appropriate equations, e.g. $reward(in(b,c) \wedge S) = 1.0$. Reward is considered to be zero for all other states.

Example 3.2. Consider action *load* from example 3.1. If a truck executing this action succeeds to load a box (supposing the truck is in the same city as the box) with a proba-

[3] The following notational conventions are introduced, unless stated otherwise: Lowercase letters represent terms (that may contain free variables), uppercase letters represent free variables. In particular, $t, t', T, \in$ TRUCK, $b, b', B \in$ BOX and $c, C \in$ CITY represent constants and free variables denoting domain objects; $s, s', s'_i \in$ STATE and $a, a' \in$ ACTION represent state and action terms, $S \in$ STATE denotes a free state variable; $p, p_i, v, v' \in \mathbb{R}$[4] denote probabilities of transitions and values of states, respectively.

[5] In order to match or unify a subject SAV-tuple with the rule's lefthand side in MAUDE, the rule has to be encoded in the form $(s \wedge S, a) \rightarrow (s' \wedge S, p)$. This encoding also explicitly shows the solution to the frame problem.

bility of 0.9, and fails to load it with a probability of 0.1, these transitions are expressed in terms of the following rewrite rule:

$$(in(T,C) \wedge in(B,C) \wedge S,\ load(T,B)) \rightarrow \\ (in(T,C) \wedge on(B,T) \wedge S,\ 0.9) \vee (in(T,C) \wedge in(B,C) \wedge S,\ 0.1)\ .$$

Note that if there were multiple types of boxes, e.g. light and heavy ones with different dynamics, these can be specified by exploiting sort-orders and polymorphism. For example, sorts LIGHT-BOX and HEAVY-BOX could be defined as subsort of sort BOX, allowing for polymorph specification of transition dynamics' rewrite rules. ▲

Note that specifying RMDPs as rewrite theories is parametric in the underlying representation of states and actions. Thus, while the example in this section treated the encoding of RMDPs as rewrite theories for a fluent-based representation, the approach can easily be transferred to other representational paradigms, e.g. OO-MDPs [8] or object-focused MDPs [7], as rewriting logic provides user-definable syntax, equational abstraction and order-sorted, polymorph specification of objects and operations [6]. Note that the MAUDE language also implements support for these features.

3.2. Symbolic Value Iteration

Regression through Narrowing. To allow for regressive induction of state-action space abstractions from given goal states, rewrite rules that specify domain dynamics are transformed into regressive rewrite rules. The key idea is to define for a given state from which preceding states it can be reached by execution of a particular action, and with what probability this action will lead to the given state. The value of the reached state is then used to compute values of preceding states according to transition probabilities.

Consider a relational MDP (S,A,T,γ,R), a value function $V : S \rightarrow \mathbb{R}$, and a rewrite theory $(\Sigma, E \cup A, R)$ encoding the MDP as outlined in section 3.1. Then, the regressive dynamics of an action can be specified by inverting the rewrite rule that specifies an effect for this action (which is of the form $(s,a) \rightarrow (s'_1,p_1) \vee ... \vee (s'_n,p_n)$) for each of the effects specified in its righthand side, i.e. for each $i \in [1,...,n]$:

$$(s'_i, V(s'_i)) \rightarrow (s,a,V(s'_i) * p_i * \gamma)\ .$$

Example 3.3. Let $V \in \mathbb{R}$ encode the value for particular abstract states, then the regressive rewrite rules for action *load* from example 3.2 are:

$$(in(T,C) \wedge on(B,T) \wedge S,\ V) \rightarrow (in(T,C) \wedge in(B,C) \wedge S,\ load(T,B),\ V * 0.9 * \gamma).$$
$$(in(T,C) \wedge in(B,C) \wedge S,\ V) \rightarrow (in(T,C) \wedge in(B,C) \wedge S,\ load(T,B),\ V * 0.1 * \gamma).$$

▲

For value iteration, this representation of domain dynamics serves two purposes. First, it allows to compute from a given value function V all abstract (i.e. relational) states from which a particular state $s' \in Domain(V)$ is reachable by narrowing (for a single step) a SAV-tuple (s',v) with $v = V(s')$ according to the inverted rewrite rules of a rewrite theory encoding a RMDP. Second, with regard to value iteration as outlined

in equation 1, this narrowing step resembles computation of $\gamma * T(s,a,s')V_i(s')$ when computing $V_{i+1}(s)$: It produces a set of SAV-tuples $(s,a,v*p*\gamma)$ that encode the action a that, when executed in state s, would result in state s' with probability p.

As narrowing is employed, variable grounding is (only) applied where necessary (i.e. where relevant to the reachable state s') through unification of the state that is currently regressed with the lefthand sides of rewrite rules specifying domain dynamics. Also, if the state to be regressed misses any action postconditions, these are induced to regressed state terms by unification if the subject term to be regressed and action effect rules' lefthand sides contain a free state variable (i.e. are of the form $s \wedge S$). Thus, also partially specified goal states can be regressed. If system goals are specified in V_0 (e.g. by instantiating V_0 with a set of SAV-tuples $(s,R(s))$, thus resembling the MDP's reward function R), narrowing exactly grounds variables and induces fluents that are relevant for an optimal policy w.r.t. these goals. Note that regression may lead to states that violate constraints (see section 3.1), which are ignored by further computation.

Summation of Non-Deterministic Action Effects. Regressing the SAV-tuples of a given value function computes a set of SAV-tuples (s,a,v) denoting the states s from which the states in the value function domain can be reached through execution of action a. While v already incorporates transition probabilities and known state values, it does not yet take into account that an action may have multiple outcomes. In equation 1, this fact is addressed by the summation of expected values of all states that are reachable by execution of a particular action a, weighted by transition probabilities. This computation can be resembled by performing a progressive one-step rewrite of $s \times a$ according to the rewrite rules from the RMDP specification (see section 3.1), and adding up the values of the resulting states s' according to V, weighted by transition probabilities. More formally, for all regressed (s,a,v) compute v by $(s \times a) \rightarrow_1 \bigvee_i (s_i \times p_i) \Rightarrow v = \sum_i (V(s_i) * p_i)$.

Maximization through Abstract State Subsumption. To ensure that only optimal actions for each abstract state remain in the new value function V_{i+1}, only those elements that exhibit the maximal value that can be gained through action execution for each possible state are kept in the set of SAV-tuples. As states are relational, they may overlap or even subsume other states completely. To deal with subsumption, the concept of Ax-matching can directly be employed to model state subsumption, as a more general term Ax-matches a more concrete one. For maximization over the set of actions, a state s' with value v' is removed from the regressed set of SAV-tuples if it is subsumed by another state s with greater or equal value: $(s,a,v) \vee (s',a',v') = (s,a,v)$ **if** $s :=_{\mathrm{Ax}} s' \wedge v \geq v'$.

Example 3.4. Let $(on(b,t) \wedge S, a, 1.0)$ and $(on(B,T) \wedge S, a', 2.0)$ be SAV-tuples in the set to be maximized. In this case, it is clearly preferable to execute action a' when *any* box is on *any* truck, because the expected value of this action is 2.0. Thus, the former tuple can be dropped. Now suppose $(on(b,t) \wedge S, a, 3.0)$ and $(on(B,T) \wedge S, a', 2.0)$ should be maximized. Then the former should not be dropped, as action a is preferable if exactly box b is on truck t; otherwise, if another box or another truck are involved, action a' should be executed. Both tuples are necessary to deduce this behaviour. ▲

Value Iteration & Decision List Policies. To complete a value iteration step according to equation 1 after performing maximization, the currently gained reward for all states in the set of SAV-tuples has to be distributed according to the MDP's reward function R.

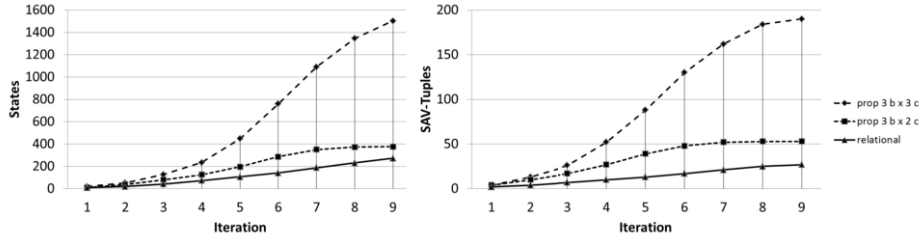


Figure 1. Number of regressed states *(left)* and value function entries *(right)* per iteration step for propositional and relational domains; reward for three boxes at destination.

New SAV-sets resembling a RMDP's value function are iteratively constructed by abstract state regression, summation of state values taking into account non-deterministic action effects, maximization of expected value for states and actions, discounting and reward distribution. For each iteration i, the resulting SAV-set exactly resembles V_{i+1} in value iteration according to equation 1 for all $s \in S$ from which states in the domain of V_i are reachable. States that are not covered by the set are assigned a value of zero, thus the SAV-set can be considered a function. Iteration stops if for all $(s,a,v) \in V_{i+1}$ there exists a $(s,a,v') \in V_i$ such that $|v - v'| < \varepsilon(1-\gamma)/\gamma$ for a given error bound $\varepsilon \in \mathbb{R}$.

The resulting SAV-set representing the converged optimal value function V (with an error bounded by ε) can then be interpreted as a decision list, sorted by values of the SAV-tuples it contains; thus, overlapping and subsuming states are dealt with. I.e., the decision list resembles a policy $\pi : S \to A$ for the MDP that was solved with the presented algorithm, considering $\pi(s) = a \Leftrightarrow (s,a,v) \in V \wedge \forall (s,a',v') \in V : v \geq v'$ and $\pi(s)$ being any action (e.g. *noop*) if $\nexists\, a, v : (s,a,v) \in V$.

4. Evaluation

The approach was experimentally evaluated with an implementation[6] of value iteration in MAUDE [6]. Experiments were conducted with the BOXWORLD domain [3], where a truck has to deliver boxes to their destination cities. A truck can either do nothing, drive from one city to another, load boxes, or unload them. Actions succeed with a probability of 0.9, and fail with no effect otherwise. Comparison was performed for a relational version of the underlying MDP (7 transition rules) and two propositional instantiations of it, one with 3 boxes and 2 cities (37 rules), the other one with 3 boxes and 3 cities (55 rules). Note that the relational rules are valid for *any* number of domain objects. This gain of scalability is not limited to specification, it holds as well for computation of solutions and for value functions themselves. Figure 1 shows the number of regressed states (and size of the value function) per iteration step for the two propositional MDPs and the relational one when a reward is specified for three boxes being in a particular city: The number of regressed states (the number of entries in the value function, respectively) grows substantially with domain size and iteration depth; even the smaller propositional MDP is outperformed by the relational one. As for the specification, the result of symbolic value iteration is valid for *any number of domain objects*. Correctness of the relational solution

[6]The implementation is available at http://www.pst.ifi.lmu.de/~belzner/odin/.

w.r.t the propositional ones was experimentally approved by checking if for each entry in the propositional value function there is a corresponding entry in the relational one.

5. Related Work & Conclusion

Related Work. The first successful approach to solve MDPs with value iteration completely on the symbolic level was achieved by *Symbolic Dynamic Programming* (SDP) [3]. It uses the situation calculus [17] to represent first-order MDPs, thus allowing for full first-order logic quantifications for variables. While the situation calculus is a very expressive specification language, the frame problem has to be addressed explicitly in the specification of domain dynamics, in contrast to specifications in rewriting logic. Another difference of SDP to the approach presented in this paper is that dynamics are defined in a regressive manner and per fluent (in terms of so-called *successor state axioms*) and not per action, diverging from modern software design paradigms as for example object-orientation, where dynamics are typically defined in terms of operations. In consequence, compilation of regressive successor state axioms from progressive, operation-oriented specifications becomes a complex transformation. Also, because of the complexity of regressed state formulas, consistency checking and simplification is a complex task. Even if these tasks are manageable automatically in theory, the authors of SDP only reported on a preliminary implementation that illustrated their approach, but simplification of results was applied manually. The *fluent calculus* [20] can be considered a progressive counterpart to the situation calculus as it represents states as associative-commutative terms of fluents. First-order value iteration for the fluent calculus (FOVIA) [11,12] can be performed in a fully automated manner due to restricted expressivity of the fluent calculus when compared to the situation calculus, as only existential quantification of variables is allowed. As the presented approach, FOVIA performs state subsumption for value function simplification and employs AC1-unification to perform regression, but it is not parametrizable in terms of state representation. In contrast to both SDP and FOVIA, using rewrite theories for symbolic value iteration leads to natural support for flexible, domain-specific representations when it comes to specification of RMDPs, allowing to include features like free choice of syntax and abstraction level, explicit sort ordering, polymorphism or object-orientation.

Summary & Further Work. This paper discussed the representation of RMDPs in rewriting logic and how to use the concepts of matching and narrowing to solve them symbolically, resulting in advantages regarding computational effort and expressivity when compared to propositional solution techniques. To this end, RMPDs were represented as rewrite theories. Regression is performed by exploiting the capabilities of narrowing, allowing for symbolic computation. Simplification of the resulting symbolic value function through state subsumption was realized by Ax-matching state terms. By relating variables in state and action terms, both state and action space are partitioned properly. The specification of RMDPs in terms of rewrite theories allows for domain-specific, user-definable representations by exploiting free syntax choice, sort-hierarchies, polymorphism and object-orientation, achieving small representational gaps between human expert knowledge and domain encoding. A clear reduction of state space and value function size was shown when comparing relational value iteration with rewrite theories to propositional value iteration. A possible direction for future research is to ex-

plore the combination of rewrite theories with algebraic representations for RMDPs (e.g. FOADDs, see [19]).

References

[1] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1957.

[2] Lenz Belzner. Verifiable decisions in autonomous concurrent systems. In Eva Kühn and Rosario Pugliese, editors, *COORDINATION*, volume 8459 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2014.

[3] Craig Boutilier, Raymond Reiter, and Bob Price. Symbolic dynamic programming for first-order MDPs. In Bernhard Nebel, editor, *IJCAI*, pages 690–700. Morgan Kaufmann, 2001.

[4] Manuel Clavel, Francisco Durán, Steven Eker, Santiago Escobar, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn L. Talcott. Unification and narrowing in maude 2.4. In Ralf Treinen, editor, *RTA*, volume 5595 of *Lecture Notes in Computer Science*, pages 380–390. Springer, 2009.

[5] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Jose F. Quesada. Maude: specification and programming in rewriting logic. *Theor. Comput. Sci.*, 285(2):187–243, 2002.

[6] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn L. Talcott, editors. *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*, volume 4350 of *Lecture Notes in Computer Science*. Springer, 2007.

[7] Luis C. Cobo, Charles L. Isbell, and Andrea Lockerd Thomaz. Object focused q-learning for autonomous agents. In Maria L. Gini, Onn Shehory, Takayuki Ito, and Catholijn M. Jonker, editors, *AAMAS*, pages 1061–1068. IFAAMAS, 2013.

[8] Carlos Diuk, Andre Cohen, and Michael L. Littman. An object-oriented representation for efficient reinforcement learning. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *ICML*, volume 307 of *ACM International Conference Proceeding Series*, pages 240–247. ACM, 2008.

[9] Steven Eker. Fast matching in combinations of regular equational theories. *Electr. Notes Theor. Comput. Sci.*, 4:90–109, 1996.

[10] Santiago Escobar, José Meseguer, and Prasanna Thati. Narrowing and rewriting logic: from foundations to applications. *Electr. Notes Theor. Comput. Sci.*, 177:5–33, 2007.

[11] Axel Großmann, Steffen Hölldobler, and Olga Skvortsova. Symbolic dynamic programming within the fluent calculus. In *Proceedings of the IASTED International conference on Artificial and Computational Intelligence*, pages 378–383, 2002.

[12] Steffen Hölldobler and Olga Skvortsova. A logic-based approach to dynamic programming. In *Proceedings of the Workshop on Learning and Planning in Markov Processes–Advances and Challenges at the Nineteenth National Conference on Artificial Intelligence (AAAI04)*, pages 31–36, 2004.

[13] John Mccarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, volume 4, pages 463–502, 1969.

[14] José Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theor. Comput. Sci.*, 96(1):73–155, April 1992.

[15] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[16] Aswin Raghavan, Saket Joshi, Alan Fern, Prasad Tadepalli, and Roni Khardon. Planning in factored action spaces with symbolic dynamic programming. In Jörg Hoffmann and Bart Selman, editors, *AAAI*. AAAI Press, 2012.

[17] Raymond Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, Massachusetts, MA, illustrated edition, 2001.

[18] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.

[19] Scott Sanner and Craig Boutilier. Practical solution techniques for first-order mdps. *Artificial Intelligence*, 173(56):748 – 788, 2009. Advances in Automated Plan Generation.

[20] Michael Thielscher. Introduction to the fluent calculus. *Electron. Trans. Artif. Intell.*, 2:179–192, 1998.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-71

On Evaluating Interestingness Measures for Closed Itemsets

Aleksey BUZMAKOV [a,b,1], Sergei KUZNETSOV [a] and Amedeo NAPOLI [b]
[a] *Higher School of Economics, Moscow, Russia*
[b] *LORIA (CNRS – Inria NGE – Université de Lorraine), France*

Abstract. There are a lot of measures for selecting interesting itemsets. But which one is better? In this paper we introduce a methodology for evaluating interestingness measures. This methodology relies on supervised classification. It allows us to avoid experts and artificial datasets in the evaluation process. We apply our methodology to evaluate promising measures for itemset selection, such as leverage and stability. We show that although there is no evident winner between them, stability has a slightly better performance.

Keywords. data mining, pattern selection, interestingness measures, stability, leverage, comparison

1. Introduction

One of the most important and frequent tasks in artificial intelligence is selection of the best option(s) among a huge set of possibilities. For example, in data mining one should often determine which patterns are of high interest. Usually patterns are evaluated w.r.t. a formal relevancy measure. Webb [1] says that measures cannot often reflect the true value of patterns because they "often depend on many factors that are difficult to formalize". *Are we able to evaluate how well a measure approximates an expert interest?*

One way to do that is to evaluate patterns with experts [2]. In that case we evaluate how close the selected patterns approximate the expert knowledge. It is an expensive strategy requiring many experts for a domain. Thus, if one wants to compare measures on datasets from different domains the experiments become very expensive. Additionally such an experimentation requires a lot of time to be carried out.

Another way to evaluate patterns is to use artificial datasets [3], where the target patterns are known. The drawback of this approach is the relevancy of the artificial datasets w.r.t. real ones. Thus, *one goal* of this paper is to develop and evaluate a methodology for comparison of interestingness measures for itemsets without involving experts or artificial datasets (below we use indifferently "pattern" or "itemste").

Our methodology *is based on* semi-supervised classification, where every data entry has a class label but labels are not directly involved into computation of a measure. A label is an additional information to entry description modeling domain knowledge or

[1]Corresponding Author: Aleksey Buzmakov, LORIA (CNRS – Inria NGE – Université de Lorraine), 615, Jardin Botanique street, 54600, Vandoeuvre-les-Nancy, France; E-mail: aleksey.buzmakov@inria.fr.

Table 1. A toy dataset

	a	b	c	d	e	f	Label
g_1	x				x	x	+
g_2		x			x	x	-
g_3			x		x	x	+
g_4				x	x	x	-
g_5		x	x			x	+
g_6		x	x		x	x	?(+)
g_7	x		x			x	?(+)
g_8	x		x	x		x	?(-)

expert intent. The basic idea is to rank patterns with an interestingness measure and, then, find among them the patterns that are relevant to classification. And if a measure $\mathscr{M}_1$ is better than another measure $\mathscr{M}_2$ w.r.t. expert interest, i.e. $\mathscr{M}_1$ attributes more systematically high ranks to more relevant patterns, thus increasing the performance of a classifier based on $\mathscr{M}_1$.

This methodology can be applied when a measure does not rely on class labels. Then it can find itemsets that are suitable for expert interest (and not biased towards the classification task).

The second goal of this paper is to evaluate some measures for itemset ranking. We evaluate leverage [4] and stability [5] measures that seem to be well adapted to itemset ranking. We also introduce difference measure that comes from an estimate of stability [6]. This measure is computed faster than stability. As it is widely used, the support of an itemset is used as a baseline measure. Finally, we also consider another leverage measure (rule leverage) which in contrast to the afore mentioned measures, relies on class labels. This rule leverage measure provides an idea of rule ranking measures w.r.t. itemset ranking measures.

Finally, we show that although there is no evident winner among stability and leverage measures, stability seems to be better on average. It is also shown that difference and stability have a similar behaviour. But according to previous studies, difference is faster to compute [6]. We can summarize *the novelty* of this paper as follows:

1. Methodology for comparison of measures for itemset ranking.
2. Comparison of leverage and stability measures for itemsets.

The rest of the paper is organised as follows. Section 2 introduces basic notions. Then related work is discussed. In Section 4 we define and discuss the evaluated measures. The next section describes our methodology. And finally before concluding the paper the experiments are discussed.

2. Preliminaries

In this section we discuss the basic definitions in terms of Formal Concept Analysis [7].

Definition 1. *A dataset or a context is a triple (G,M,I), where G is a set of objects, M is a set of attributes, and $I \subseteq G \times M$ is a relation between G and M.*

Every subset of attributes is called *a pattern* or *an itemset*. *The description* of a set of objects X is the set of attributes shared by all objects from X, $\phi(X) = \{m \in M \mid (\forall g \in X) gIm\}$. *The image* of itemset Y is the set of objects sharing Y, $\psi(Y) = \{g \in G \mid (\forall m \in Y) gIm\}$. The cardinality of the image of Y is called *support* of Y, $\mathtt{Supp}(Y) = |\psi(Y)|$, while the value $\sigma(Y) = \frac{\mathtt{Supp}(Y)}{|G|}$ is called *frequency* of Y.

Consider the toy dataset in Table 1. Let $G = \{g_1, g_2, g_3, g_4, g_5\}$, $M = \{a, b, c, d, e, f\}$ and I is shown in the table, then $\phi(\{g_1, g_2\}) = \{e, f\}$ is the set of attributes shared by g_1 and g_2 or the description of the set $\{g_1, g_2\}$. Similarly $\psi(\{e, f\}) = \{g_1, g_2, g_3, g_4\}$ is the image of $\{e, f\}$. We say that the support of the itemset $\{e, f\}$ is $\mathtt{Supp}(\{e, f\}) = 4$ and its frequency is $\sigma(\{e, f\}) = \frac{4}{|G|} = 0.8$.

Definition 2. *An itemset X is closed if and only if there is no superset $Y \supset X$ such that $\mathtt{Supp}(Y) = \mathtt{Supp}(X)$.*

It means that an itemset X is closed if it is not possible to add any attribute to X preserving the set of objects that supports X. The operator $\phi \circ \psi$ is a closure operator and thus an itemset X is closed if and only if $\phi(\psi(X)) = X$.

Definition 3. *An association rule between an itemset X and an itemset Y is denoted by $X \to Y$, where X is called the premise and Y is called the conclusion of the rule.*

Rule $X \to Y$ means that if the description of some objects from G contains X, then it contains Y. There are two measures attached to an association rule: support (or frequency) and confidence.

Definition 4. *The support of a rule $X \to Y$ is $\mathtt{Supp}(X \cup Y)$ and frequency of the rule $X \to Y$ is $\sigma(X \cup Y)$.*

Definition 5. *The confidence of a rule $X \to Y$ is $\mathtt{Conf}(X \to Y) = \frac{\mathtt{Supp}(X \cup Y)}{\mathtt{Supp}(X)}$.*

The support and the frequency of a rule show how often one can find the premise in the dataset, while a rule $X \to Y$ with a high confidence means that in most of the cases if an object description includes X it is likely to include Y. For example, in the dataset in Table 1 with the set of objects $G = \{g_1, g_2, g_3, g_4, g_5\}$ the confidence of the rule $\{e\} \to \{f\}$ is 1 because in every case when an object description contains e it contains also f, while $\mathtt{Conf}(\{c\} \to \{e, f\}) = \frac{1}{2}$.

A common objective in data mining is search for interesting patterns, i.e. for interesting itemsets or rules, that are usually related to a task. Among those different tasks, there are classification, clustering and expert analysis of the result. Here we focus on searching for patterns that are likely to be interesting to an expert. In the next section we describe the existing approaches for mining interesting itemsets.

3. Related Works

Probably, the most elaborated area of mining interesting patterns is association rule mining. Most of the measures created in this area optimize a formal criterion using statistical methods [8]. Although there is a huge number of interestingness measures, there are only few comparisons between them [2,9]. The main reasons for that are the diversity of

formal criteria and the fact that no measure wins in all criteria. In [2] the authors evaluate different measures by means of expert interest. This is an important approach for pattern evaluation as it directly measures the relation between a formal measure and an expert interest. The drawback of this approach is the cost and diversity of datasets: for every dataset it is necessary to hire several experts which is costly. In [9], the authors evaluate measures by their performance in the classification task in a supervised setting, e.g. how confident is a rule concluding on a given class. In such case, the aim of an expert is expressed by labeling of a training set. This is in contrast with our approach which follows a semi-supervised setting, i.e. measures do not depend on labelling.

Another group of interestingness measures consists of measures created for itemset ranking. It is a less studied group. Several measures can be found in [3]. Some of them are related to the distribution of partitions induced by every attribute from the considered pattern. Others are related to measures of association rule mining. Another approach introduces the measure of leverage that corresponds to the difference between frequency of an itemset and the maximal expected frequency based on subsets of the itemset [4]. Finally, some measures can be found in the domain of formal concept analysis [10,11, 12]. Stability measure is one of the most interesting among them, because it is often used in domain specific areas where experts are often involved. Moreover, in contrast to all above mentioned measures for itemsets, stability is computed on object side making it possible to apply it for ranking any types of patterns, e.g. sequential patterns [13].

One comparison of interestingness measures of itemsets can be found in [3] where the authors introduce a Quest Generator, i.e. a tool generating a dataset from a given set of "goal" itemsets in the presence of possible noise. Then, the interestingness measures can be evaluated w.r.t. their ability for finding the "goal" itemsets. For all artificial tests there is always a question about the degree to which generated datasets reflect real data. Thus, in this paper we provide an alternative approach for evaluating interestingness measures of itemsets on real datasets without involving experts. In the next sections we consider and compare these measures in details.

4. Itemset Interestingness Measures

For comparison the stability and leverage measures are selected as the most recent and promising measures. Support is also included into the comparison as a baseline measure, since it is widely used in itemset mining.

Definition 6 ([11]). *Given a context (G,M,I), the stability of an itemset $Y \in 2^M$ is given by the following formula:*

$$\mathtt{Stab}(Y) = \frac{|\{X \subseteq \psi(Y) \mid \phi(X) = Y\}|}{2^{\mathtt{Supp}(Y)}}. \tag{1}$$

The intuition behind stability is the following [11]. Stability gives an idea of how much a closed itemset depends on an object in its image, i.e. if we remove an object does the closed itemset exists anymore? In other words, given a dataset of objects and an itemset, stability measures the probability that the same itemset can be found in a dataset built as a subset of the objects. Consider for example itemset $\{e,f\}$ for the dataset in Table 1. The image of this set is $\{g_1,g_2,g_3,g_4\}$ when $G = \{g_1,g_2,g_3,g_4,g_5\}$. There are

16 possible subsets of this image. The descriptions of the $\emptyset$, $\{g_1\}$,$\{g_2\}$,$\{g_3\}$,$\{g_4\}$, and $\{g_5\}$ are different from $\{e,f\}$. Then, stability of $\{e,f\}$ is $\mathtt{Stab}(\{e,f\}) = \frac{16-5}{16} = 0.69$. Similarly the stability of $\{d,e,f\}$ is 0.5. According to the definition, stability of a non-closed itemset is always 0.

Although, stability is a measure that is hard to compute [10], it can be efficiently estimated [6]: $\mathtt{Stab}(Y) \leq 1 - 2^{-(\mathtt{Supp}(Y)-\mathtt{Supp}(X))}$, for all $X \supset Y$. Thus, stability allows us to introduce the related measure of "minimal positive difference in support" between itemset Y and itemsets including Y:

$$\mathtt{Diff}(Y) = \min_{X \supset Y,\ \mathtt{Supp}(X) \neq \mathtt{Supp}(Y)} (\mathtt{Supp}(Y) - \mathtt{Supp}(X)). \tag{2}$$

Difference can be computed efficiently and the experiments show the interestingness of this measure. For example, difference of itemset $\{e,f\}$ is 3, because support of $\{e,f\}$ is 4 and any superset of $\{e,f\}$ has support at most 1. Similarly difference of $\{d,e,f\}$ is 1. For non-closed itemsets, difference is always zero.

The next measure that we evaluate is leverage for itemsets. For defining leverage we recall that a 2-partition of a set Y is a partition of Y in two subsets V and W and is denoted by $\mathtt{Part}_2(Y) = (V|W)$. For example, the pair $(\{a,b,c\},\{e,f\})$ is a 2-partition of the set $\{a,b,c,e,f\}$. Now we can define what the leverage of an itemset is.

Definition 7 ([4]). *The leverage of an itemset $Y \in 2^M$ is the difference between $\sigma(Y)$ and the maximal frequency that would be expected under assumption of independence of any subset of Y:*

$$\mathtt{Lev}(Y) = \sigma(Y) - \operatorname*{argmax}_{(V|W)=\mathtt{Part}_2(Y)} \sigma(V)\cdot\sigma(W), \tag{3}$$

where $\mathtt{Part}_2(Y)$ is a 2-partition of Y.

According to the definition, leverage of an itemset can be applied to any itemset. If an itemset is non-closed then the leverage value is not zero and the next proposition holds.

Proposition 1. *The leverage of an itemset is not larger than the leverage of its closure,* $\mathtt{Lev}(Y) \leq \mathtt{Lev}(\phi(\psi(Y)))$.

Proof. Frequency can only decrease with addition of an attribute, i.e. $(\forall X \subseteq Y)\sigma(X) \geq \sigma(Y)$. Frequencies of an itemset and its closure are equal, $\sigma(Y) = \sigma(\phi(\psi(Y)))$. Given itemset X, a 2-partition of its closure $\mathtt{Part}_2(\phi(\psi(X))) = (V|W)$ induces the 2-partition of X, i.e. $(V \cap X | W \cap X)$ is a 2-partition of X. Then,

$$\begin{aligned}
\mathtt{Lev}(\phi(\psi(Y))) &= \sigma(\phi(\psi(Y))) - \operatorname*{argmax}_{(V|W)=\mathtt{Part}_2(\phi(\psi(Y)))} \sigma(V)\cdot\sigma(W) = \\
&= \sigma(Y) - \operatorname*{argmax}_{\substack{(V|W)=\mathtt{Part}_2(Y) \\ (P|Q) \\ \mathtt{Part}_2(\phi(\psi(Y))\setminus Y)}} \sigma(V\cup P)\cdot\sigma(W\cup Q) \geq \\
&\geq \sigma(Y) - \operatorname*{argmax}_{(V|W)=\mathtt{Part}_2(Y)} \sigma(V)\cdot\sigma(W) = \mathtt{Lev}(Y).
\end{aligned}$$

Thus, leverage maximizes its value on closed itemsets, and, consequently, we can compute it only for closed itemsets. □

Let us consider an example. In order to compute leverage of itemset $\{e,f\}$ we need to find all its 2-partitions. There is only one 2-partition $(\{e\}|\{f\})$. The frequencies are $\sigma(\{e,f\})=0.8$, $\sigma(\{e\})=0.8$, $\sigma(\{f\})=0.8$. Thus, $\texttt{Lev}(\{e,f\})=0.8-0.8^2=0.16$. For itemset $\{d,e,f\}$ we have three 2-partitions: $(\{e\}|\{d,f\})$, $(\{d\}|\{e,f\})$ and $(\{f\}|\{e,d\})$. The frequencies are $\sigma(\{d,e,f\})=0.2$, $\sigma(\{e\})\cdot\sigma(\{d,f\})=0.8\cdot 0.2=0.16$, $\sigma(\{d\})\cdot\sigma(\{e,f\})=0.2\cdot 0.8=0.16$, $\sigma(\{f\})\cdot\sigma(\{d,f\})=0.8\cdot 0.2=0.16$. Thus, $\texttt{Lev}(\{d,e,f\})=0.2-0.16=0.04$.

The leverage of an itemset is based on the notion of leverage of a rule. Hereafter, we use leverage of a rule in our comparison as a base line and, thus, we need to provide its definition.

Definition 8. *The leverage of a rule is defined as follows*

$$\texttt{Lev}(X\to Y)=\sigma(X\cup Y)-\sigma(X)\cdot\sigma(Y) \tag{4}$$

In this work rule leverage is applied to rules of the form $X\to\{\mathscr{C}\}$, where $\mathscr{C}$ is a class label in classification. Let us consider Table 1, where the target class is given by column "class". In order to define rule leverage of $\{e,f\}\to\{+\}$, first, we should find the frequencies: $\sigma(\{e,f,+\})=0.6$, $\sigma(\{e,f\})=0.8$, $\sigma(\{+\})=0.6$. Thus, $\texttt{Lev}(\{e,f\}\to\{+\})=0.6-0.8\cdot 0.6=0.12$.

We are now ready to introduce our methodology.

5. Evaluation Methodology

In this work the classification task is used to estimate the interestingness of measures for itemset selection w.r.t. expert interest, by measuring the precision and recall of classifiers built with these measures.

The intuition behind the usage of classification for evaluating measures is the following. If an itemset is of high interest for an expert, then it should reflect basic dependencies in a domain. Thus, the performance of this itemset in classification should be better than an arbitrary itemset. Consequently, systematic good performances may mean that a measure is more suitable to find itemsets of high interest. Accordingly, the evaluation methodology consists of the following steps:

1. A dataset $\mathscr{D}$ is selected.
2. The dataset $\mathscr{D}$ is divided into training and test sets by random sampling 100 times. A training set contains 90% of the objects with class labels (but at most 1000 objects which is a limit of `Magnum Opus` demo [14] that is used for leverage computation). The test set contains the rest of the objects.
3. One target class label $\mathscr{C}$ is selected.
4. One target measure $\mathscr{M}$ is selected.
5. A training set built at step 2 is used to find itemsets and rank them w.r.t. the measure $\mathscr{M}$. However, during the search, class labels for objects are ignored.

6. Among the whole set of itemsets, the emerging patterns for class $\mathscr{C}$ are selected from the training set [15]. An emerging pattern is an itemset that is a characteristic of one class, i.e. it covers objects mostly labelled with the same class, w.r.t. a threshold θ. These emerging patterns are assumed to be good for classification purposes. The idea of emerging patterns is borrowed from [16], where emerging patterns are called hypotheses. Let say that there are N emerging patterns.
7. From these N emerging patterns we form N classifiers based on the first k patterns (with $k \leq N$). Each classifier works in the following way. Given a set of patterns $\{p_1, \cdots, p_k\}$, the classifier attaches the label $\mathscr{C}$ to any object whose description contains p_i for $i \in [1,k]$.
8. We compute precision and recall for these N classifiers in the test set. Then we interpolate 21 points of the form (p,r) where p stands for precision and r stands for recall, where $r \in \{0, 0.05, \cdots, 0.95\}$. These 21 points yield a curve.
9. Steps 6–8 are repeated for every pair of training and test sets. An average curve is computed for all the curves based on the pairs of training and test sets.
10. The area under this averaged curve is computed providing a numerical quality of the measure $\mathscr{M}$ in dataset $\mathscr{D}$ w.r.t. class $\mathscr{C}$.
11. We repeat steps 3–10 for all classes in $\mathscr{D}$ and all measures.
12. We repeat steps 1–11 for all available datasets.

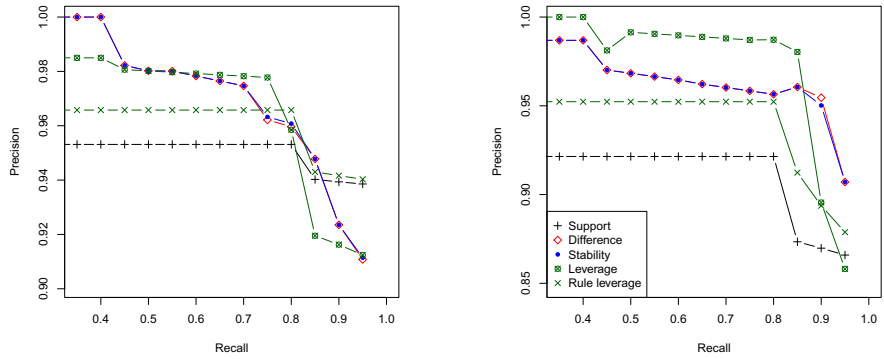
Thus, each measure is evaluated for every class label and for any division of a dataset. The precision and recall in step 8 are computed in a standard way, i.e. in terms of true/false positives/negatives where the precision is $\mathtt{Pr} = \frac{\mathtt{TP}}{\mathtt{TP+FP}}$ and the recall is $\mathtt{R} = \frac{\mathtt{TP}}{\mathtt{TP+FN}}$.

But how can one select the threshold θ? This is a tricky question. On the one hand, it is necessary to take the high θ in order to force a measure to select itemsets relevant for the classification. Thus, datasets where there are no patterns with high θ are not adapted for the methodology. On the other hand, it is necessary to have a sufficient number of emerging patterns to capture differences between measures. Here, we posed $\theta = 90\%$, i.e. at least 90% of objects in the image of a pattern are in the same class. However, the selection of an ideal θ is still an open question.

In [9] measures rely on class labeling and thus they are biased for classification task. In contrast in our work measures evaluate itemsets and after that a labeling is introduced. Thus, our approach appears to be closer to the expert interest.

Let us consider this methodology on the example in Table 1. We have a dataset containing 8 objects (step 1). This dataset is divided into training set, $Tr = \{g_1, g_2, g_3, g_4, g_5\}$, and test set $T = \{g_6, g_7, g_8\}$ (step 2). The target class label is $\mathscr{C} = +$ (step 3). The target measure is difference (step 4). In this example we consider an itemset to be an emerging pattern if 50% of objects in its image are labeled with the target class. Thus, we have five closed emerging patterns: $\{e,f\}$, $\{c,f\}$, $\{a,e,f\}$, $\{c,e,f\}$ and $\{b,c,f\}$ (step 6). The corresponding differences are 3, 1, 1, 1, 1. Thus, they are well sorted and we are ready to construct classifiers (step 7) and evaluate their performance (steps 8 and 9).

The first one is only based on $\{e,f\}$. This itemset is only included in the description of g_6, consequently only g_6 should be classified positively. The precision and recall of this classifier are 1 and 0.5. The next classifier is based on $\{e,f\}$ and $\{c,f\}$. The description of g_6 includes $\{e,f\}$ and, thus, it should be classified positively with the second classifier. The descriptions of g_7 and g_8 include $\{c,f\}$ and, thus, they should be also classified positively. The precision and recall of the second classifier is $\frac{2}{3}$ and 1. After



(a) Eatable mushrooms

(b) Poisonous mushrooms

Figure 1. Precision and Recall for mushroom dataset for classifiers built with different interestingness measures.

repeating this with all emerging patterns we can interpolate the value of precision for every recall of the form $0.05 \cdot K$, where $K \in \{1, 2, \cdots, 20\}$. Doing this several time for every division of the dataset we can obtain the averaged precisions corresponding to these recalls. Finally, we can compute the area under the average curves providing a numerical quality of the measure on this dataset.

Finally, any emerging pattern X for class $\mathscr{C}$ can be written as an association rule $X \to \{\mathscr{C}\}$. Thus, it is also possible to introduce the interestingness measures for rules in this framework as a baseline for evaluating interestingness measures of itemsets. We decided to add the leverage interestingness measure for rules, see Eq. (4). The results for rule leverage measure are provided only as a baseline because it uses the target class in the computation procedure and, thus, can be better adapted for classification purposes.

6. Experiment

The experiments are carried out with public available datasets from UCI [17]: `Mushroom`[2], `Congressional Voting Records`[3], `Nursery`[4] datasets. All datasets contain emerging patterns and thus we can apply our methodology. In the experiments we have compared 4 interestingness measures for itemset ranking, i.e. support, stability (1), difference (2) and leverage (3), as well as a measure for association rule ranking, i.e. rule leverage (4). Comparison of the computational efficiency is not the goal of this paper. Thus, we only mention that computations take less than a minute per experiment in every case.

Let us consider one dataset deeper. Figure 1 shows the results of two experiments on `Mushroom` dataset. Figure 1a shows precision and recall for predicting the class of edible mushrooms, while Figure 1b corresponds to poisonous mushrooms. Every line in

[2] `http://archive.ics.uci.edu/ml/datasets/Mushroom`
[3] `http://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records`
[4] `http://archive.ics.uci.edu/ml/datasets/Nursery`

Table 2. The surface under the ROC-diagram for different datasets different target classes and different measures. The best measure in a row is bolded.

Dataset	Class	Support	Difference	Stability	Itemset Lev.	Rule Lev.
Mushroom	Poisonous	0.890658	0.945881	0.945665	**0.956895**	0.919898
Mushroom	Eatable	0.927239	0.953793	**0.953941**	0.946683	0.938007
Vote	Democrat	0.865279	0.862507	0.8645	0.904433	**0.953708**
Vote	Republican	0.883406	**0.921093**	0.921004	0.884818	0.883406
Nursery	Not Recommended	**0.975**	**0.975**	**0.975**	**0.975**	**0.975**
Nursery	Priority	**0.78503**	0.743039	0.725221	0.605405	0.525
Nursery	Special Priority	**0.875556**	0.850174	0.851127	0.699788	0.639793

this figure corresponds to a measure. Every point corresponds to a precision-recall pair at the end of step 9 of the proposed methodology.

In this figure we can see that stability and difference have nearly the same behaviour. It is the case for every tested dataset. The second point is that the support measure is not the best one for pattern selection, which is not surprising. The unexpected result here is that the rule leverage does not perform well. Logically it should be the best one because it is the only tested measure that can access the label information in the dataset. One explanation can be that the statistical significance (at least of the rule leverage type) is not directly related to the relevancy of an itemset to real patterns.

In Table 2 the numerical qualities for every dataset and every class label is given. Every column corresponds to a measure and every line corresponds to a combination of a dataset and a class label. First, the difference and stability measures have a similar behaviour, and the numerical quality has nearly the same value in every experiment. Second, although there is no evident winner between stability and itemset leverage and they often have a comparable result, but on `Nursery` dataset stability has a significantly better result.

7. Conclusion

In this paper we have proposed a methodology for evaluating interestingness measures for closed itemset selection. The proposed methodology has been applied to compare leverage, stability and difference measure. Although stability has a slightly better behaviour than leverage we cannot conclude that one is better than the other. It is also shown that stability and difference have very similar behaviours, but difference is computed faster.

It should be noticed that stability and difference have an important property that they can be applied to any kind of datasets as soon as support can be computed, e.g. datasets of sequences or graphs. This is not, for example, the case for leverage. Since difference is faster to compute, we should conclude that difference is the most convenient measure providing the same quality as stability and leverage.

There are several directions for future research. First, other measures and other datasets should be involved into comparison for more reliable results. Second, the correlation of ranking w.r.t. different measures should be studied. Finally, since stability and leverage have the best performances on different datasets, it can be an important task to develop a powerful measure based on both approaches.

Acknowledgements

This research received funding from the Basic Research Program at the National Research University Higher School of Economics (Moscow; Russia) and from the BioIntelligence project (France).

References

[1] Geoffrey I Webb and Songmao Zhang. K-Optimal Rule Discovery. *Data Min. Knowl. Discov.*, 10(1):39–79, 2005.

[2] Deborah R. Carvalho, Alex A. Freitas, and Nelson Ebecken. Evaluating the Correlation Between Objective Rule Interestingness Measures and Real Human Interest. In Alípio Mário Jorge, Luís Torgo, Pavel Brazdil, Rui Camacho, and João Gama, editors, *Knowl. Discov. Databases PKDD 2005*, volume 3721 of *Lecture Notes in Computer Science*, pages 453–461. Springer Berlin Heidelberg, 2005.

[3] Albrecht Zimmermann. Objectively evaluating interestingness measures for frequent itemset mining. In *Emerg. Trends Knowl. Discov. Data Mining-PAKDD 2013 Int. Work.*, 2013.

[4] Geoffrey I. Webb. Self-sufficient itemsets. *ACM Trans. Knowl. Discov. Data*, 4(1):1–20, January 2010.

[5] Sergei O. Kuznetsov. On stability of a formal concept. *Ann. Math. Artif. Intell.*, 49(1-4):101–115, 2007.

[6] Aleksey Buzmakov, Sergei O Kuznetsov, and Amedeo Napoli. Scalable Estimates of Concept Stability. In Christian Sacarea, Cynthia Vera Glodeanu, and Kaytoue Mehdi, editors, *Form. Concept Anal.*, volume 8478 of *Lecture Notes in Computer Science*, pages 161–176. Springer Berlin Heidelberg, 2014.

[7] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1st edition, 1999.

[8] Adnan Masood and Stephen Soong. Measuring Interestingness – Perspectives on Anomaly Detection. *Comput. Eng. Intell. Syst.*, 4(1):29–40, 2013.

[9] Paulo J. Azevedo and Alípio M. Jorge. Comparing Rule Measures for Predictive Association Rules. In Joost N. Kok, Jacek Koronacki, Ramon Lopez de Mantaras, Stan Matwin, Dunja Mladenič, and Andrzej Skowron, editors, *Mach. Learn. ECML 2007*, volume 4701 of *Lecture Notes in Computer Science*, pages 510–517. Springer Berlin Heidelberg, 2007.

[10] Sergei O. Kuznetsov. Stability as an Estimate of the Degree of Substantiation of Hypotheses on the Basis of Operational Similarity. *Autom. Doc. Math. Linguist. (Nauch. Tekh. Inf. Ser. 2)*, 24(6):62–75, 1990.

[11] Camille Roth, Sergei Obiedkov, and Derrick G Kourie. On succinct representation of knowledge community taxonomies with formal concept analysis A Formal Concept Analysis Approach in Applied Epistemology. *Int. J. Found. Comput. Sci.*, 19(02):383–404, April 2008.

[12] Radim Belohlavek and Martin Trnecka. Basic Level in Formal Concept Analysis: Interesting Concepts and Psychological Ramifications. In *Proc. Twenty-Third Int. Jt. Conf. Artif. Intell.*, IJCAI'13, pages 1233–1239. AAAI Press, August 2013.

[13] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Data Eng. 1995. Proc. Elev. Int. Conf.*, pages 3–14, March 1995.

[14] Geoffrey I. Webb. Discovering Significant Patterns. *Mach. Learn.*, 68(1):1–33, 2007.

[15] Guozhu Dong and Jinyan Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proc. fifth ACM SIGKDD Int. Conf. Knowl. Discov. data Min.*, KDD '99, pages 43–52, New York, 1999. ACM.

[16] Sergei O. Kuznetsov. Mathematical aspects of concept analysis. *J. Math. Sci.*, 80(2):1654–1698, 1996.

[17] A. Frank and A. Asuncion. *UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]*. University of California, Irvine, School of Information and Computer Sciences, 2010.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-81

Learning Probabilistic CP-nets from Observations of Optimal Items

Damien BIGOT [a], Jérôme MENGIN [a] and Bruno ZANUTTINI [b]
[a] *IRIT, Université Paul Sabatier, Toulouse, France*
[b] *GREYC, Université de Caen Basse-Normandie, France*

Abstract. Modelling preferences has been an active research topic in Artificial Intelligence for more than fifteen years. Existing formalisms are rich and flexible enough to capture the behaviour of complex decision rules. However, for being interesting in practice, it is interesting to learn not a single model, but a probabilistic model that can compactly represent the preferences of a group of users – this model can then be finely tuned to fit one particular user. Even in contexts where a user is not anonymous, her preferences can depend on the value of a non controllable state variable. In such contexts, we would like to be able to answer questions like "What is the probability that o is preferred to o' by some (unknown) agent?", or "Which item is most likely to be the preferred one, given some constraints?"

We study in this paper how Probabilistic Conditional Preference networks can be learnt, both in off-line and on-line settings.

Keywords. PCP-net, Learning, preference, recommandation

1. Introduction

The development of recommender systems and other interactive systems for supporting decision-making has highlighted the need for models capable of using a user's preferences to guide her choices. Modelling preferences has been an active research topic in Artificial Intelligence for more than fifteen years. In recent years, several formalisms have been proposed that are rich enough to describe in a compact way complex preferences of a user over combinatorial domains. When the user's preferences are qualitative, and have a "simple" structure, conditional preference networks (CP-nets, [5]) and their variants [4,6] are popular representation frameworks. In particular, CP-nets come with efficient algorithms for finding most preferred items (*item optimisation problem*).

Existing formalisms are rich and flexible enough to capture the behaviour of complex decision rules. However, for being interesting in practice, these formalisms must also permit fast elicitation of a user's preferences, involving a reasonable amount of interaction only. Anonymous recommendation systems, preference-based search [17], or configuration of combinatorial products in *business-to-customer* problems [15] are good examples of decision problems in which the user's preferences are not known *a priori*. In such applications, a single interaction with the user must typically last at most 0.25 s, and the whole session must typically last at most 20 minutes, even if the item to be recommended to the user is searched for in a combinatorial set.

Recently there have been several interesting proposals for learning preferences, many of them where presented in [11]. The approaches range from learning numerical ranking functions [12,18,1] to learning qualitative, structured preference rules [13,10,14,2]. These works assume that a set of rankings or pairwise comparisons is given or elicitated, in order to build a model that generalises these rankings or comparisons.

However, in several settings, it is interesting to learn not a single model, but a probabilistic model that can compactly represent the preferences of a group of users. In a next step, this model can then be finely tuned to fit one particular user. Even in contexts where a user is not anonymous, her preferences are usually ill-known, because they can depend on the value of a non controllable state variable. In such contexts, we would like to be able to answer questions like "What is the probability that o is preferred to o' by some (unknown) agent?", or "Which item is most likely to be the preferred one, given some constraints?"

Probabilistic Conditional Preference networks (or PCP-nets for short) [9,8] enable the user to compactly represent a probability distribution over some partial orderings and answer such queries. Specifcially, a PCP-net specifies a probability distribution over a family of CP-nets. There is a close connection between CP-nets and Bayesian networks: [7] proves that the problem of finding the most probable optimal item is similar to an optimisation problem in a Bayesian network. However, a PCP-net encodes a probability distribution over partial orders, not just on a list of items.

We study in this paper how PCP-nets can be learnt, both in off-line and on-line settings. Appart from the probabilistic approach, one difference with the works mentioned above is that we do not assume that we have, or elicitate, a set of rankings or pairwise comparisons. Instead, we suppose that we have a list of items which, it is assumed, are or have been optimal for some user or in some context. Such a list can be, for instance, a list of items that have been sold. We prove that such information is sufficient to learn a partial order over the set of possible items, when these have a combinatorial structure.

The elicitation of probabilistic CP-nets is discussed by [16]. However, the authors did not give a precise semantics to their CP-nets.

The next section sums up the main properties of CP-nets and PCP-nets. We then describe how it is possible to learn PCP-nets, off-line and on-line. Finally, we show the results of some experiments that simulate an on-line learning setting.

2. Background on probabilistic CP-nets

We consider combinatorial objects defined over a set of n variables $\mathcal{V}$. Variables are denoted by uppercase letters $A,B,X,Y,\ldots$. In this paper, we suppose that variables are Boolean; we consistently write x and $\bar{x}$ for the two values in the domain $\underline{X}$ of X.

For a set of variables $U \subseteq \mathcal{V}$, $\underline{U}$ denotes the Cartesian product of their domains. Elements of $\underline{\mathcal{V}}$ are called *items*, denoted by $o,o',\ldots$. Elements of $\underline{U}$ for some $U \subseteq \mathcal{V}$ are denoted by $u,u',\ldots$. Given two sets of variables $U,V \subseteq \mathcal{V}$ and $v \in \underline{V}$, we write $v[U]$ for the restriction of v to the variables in $U \cap V$.

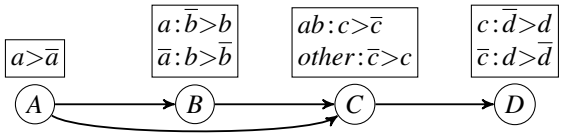
Preferences are encoded in CP-nets using rules of the form $X,u:>$, where $X \in \mathcal{V}$, u is an instantiation of variables in a set $U \subseteq \mathcal{V}$ that does not contain X, and $>$ is a total strict order over the domain of X: either $x>\bar{x}$ or $\bar{x}>x$. Informally, the rule $X,u:x>\bar{x}$ can

be read:"Whenever u is the case, then x is preferred to $\bar{x}$, *ceteris paribus* (all other things being equal)."

A rule $X,u:>$, with $u \in \underline{U}$, indicates that the preference over the values of X depends on the values of the variables in U. Associated to every CP-net is a directed graph G over $\mathscr{V}$: there is an edge (Y,X) whenever the preference over the values of X depends on the values of Y; G is called the *structure* of the CP-net. We write $\mathsf{pa}(X)$ for the set of variables on which the order over $\underline{X}$ depends, called the parents of X: $\mathsf{pa}(X) = \{Y \in \mathscr{V} \mid (Y,X) \in G\}$. It is generally assumed that a CP net contains a rule $(X,u:>)$ for every $X \in \mathscr{V}$ and every $u \in \mathsf{pa}(X)$; the set of the rules that order the domain of X is called the *conditional preference table*, or CPT, for X. When X is clear from the context, we write $u:>$ instead of $(X,u:>)$ A CP-net specifies a partial ordering $\succ$ over the set of items: $\succ$ is the transitive closure of the set of the pairs of items (o,o') such that there is a rule $(X,u:>)$ with $o[X] > o'[X]$ and $o[U] = o'[U]$ and $o[Y] = o'[Y]$ for every $Y \notin (U \cup \{X\})$. When needed, we will distinguish the partial ordering associated with a particular CP-net N using a subscript: $\succ_N$.

CP-nets are most interesting when there is no cyclic preferential dependency between the variables, that is, when the graph G does not contain any (directed) cycle. In this case, [5] proved that the relation $\succ$ is a (partial) strict order.

Example 1 *An example of an acyclic CP-net over 4 binary variables A,B,C,D is:*



The rule $ab : c > \bar{c}$ implies that $abcd \succ ab\bar{c}d$. We also have that $ab\bar{c}d \succ ab\bar{c}\bar{d}$ because of the rule $\bar{c} : d > \bar{d}$, thus, by transitivity, $abcd \succ ab\bar{c}\bar{d}$.

Optimization can be done in time linear in the size of an acyclic CP-net: choose an ordering $X_1, \ldots, X_n$ of the variables in $\mathscr{V}$ that is compatible with the dependency graph (if $X_i \in \mathsf{pa}(X_j)$ then $i < j$), and assign in turn to every X_j its most preferred value, given the values that have already been chosen for its parents. The resulting item is the unique optimal (undominated) item of the ordering specified by the CP-net. For instance, the order represented by the CP-net above has exactly one optimal item, which is $ab\bar{c}d$ (ie there is no object which is preferred to $ab\bar{c}d$).

The *forward sweep* procedure above can also be used to find an item that is optimal among those that satisfy a given conjunction of constraints of the form $Y_k = y_k$ – one only has to find, for each X_i, the optimal admissible value given the value of its parents. For instance, the most optimal item, among those that have b as value for B, is $ab\bar{c}d$. Conversely, if we know the structure of a CP-net and the optimal item o, then we can immediately induce some of the rules of the CP-net: for every $X \in \mathscr{V}$, the CP-net contains the rule $(X, o[\mathsf{pa}(X)] : o[X] > \overline{o[X]})$.

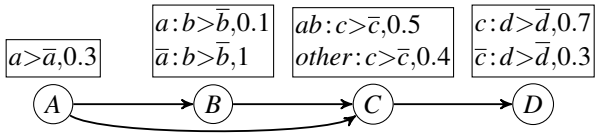
Note that the *dominance* problem, that is deciding, given a CP-net and two items o and o', if $o \succ o'$, is an NP-hard problem, even for acyclic CP-nets. Yet, a weaker *ordering query* can be answered in time linear in the number of variables as follows: given an acyclic CP-net and two items o and o', choose again an ordering $X_1, \ldots, X_n$ that is compatible with G, and consider the variables one after the other as long as $o[X_i] = o'[X_i]$; let

now i be the first i such that $o[X_i] \neq o'[X_i]$: if $o[X_i] > o'[X_i]$ (resp. $o[X_i] < o'[X_i]$) given the values of the parents of X_i in o and o', then $o' \not\succ o$ (resp. $o \not\succ o'$).

2.1. Probabilistic CP-nets

Uncertainty about the ordering over the items can be represented by associating probabilities to a given preference structure G [9,8]: for each pair (X,u), with $X \in \mathcal{V}$ and $u \in \mathsf{pa}(U)$ a probability distribution is defined over the set of possible orderings over $\underline{X}$; in the case of boolean variables, this distribution is entirely defined by the probability that the ordering is $x > \overline{x}$. In the sequel, we write $p(X,u : x > \overline{x})$ for this probability. A *probabilistic CP-net*, or PCP-net for short, is entirely defined by its structure G and the probabilities $p(X,u : x > \overline{x})$.

Example 2 *A probabilistic CP-net with the same structure as the CP-net of Example 1:*



Suppose that a PCP-net is given, with structure G. Assuming that the orderings over the domains of the variables are probabilistically independent from one another, the probability that a given CP-net N, with the same structure G, occurs, can be defined as:

$$P(N) = \prod_{(X,u)} p(X,u : >^N_{X,u})$$

where the product is taken over all variables $X \in \mathcal{V}$ and assignments $u \in \mathsf{pa}(X)$, and where $>^N_{X,u}$ denotes the ordering over $\underline{X}$ that occurs in N when u is the case.

So, a PCP-net $\mathcal{N}$ is not intended to represent a preference relation. Rather, it represents a probability distribution over a set of CP-nets, namely, those which have the same structure as $\mathcal{N}$: we say that they are *compatible* with the PCP-net. We will write $N \propto \mathcal{N}$ to indicate that the CP-net N has the same structure as $\mathcal{N}$.

Note that, in a PCP-net, the probabilities of the rules are independent from one another. So, for instance, it would not be possible to represent with a PCP-net a probability distribution over CP-nets with the structure of Example 2 if the rules over B and C were dependent; if the preferred values for B and C were always b and c together, or $\overline{b}$ and $\overline{c}$. An important topic for further research is to generalize the approach to allow for such dependencies.

Given a PCP-net $\mathcal{N}$, which represents a probability distribution on a set of deterministic CP-nets, reasoning tasks consist in computing probabilities associated with interesting events.

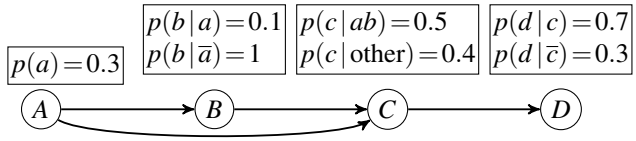
2.1.1. Probabilistic optimization

Let, for any item o, "opt$=o$" denote the set of compatible CP-nets that have o as unique optimal item. Then $P(\text{opt}=o)$ is, by definition, the sum of the probabilities $P(N)$ of the CP-nets N that are compatible with $\mathcal{N}$ and such that o is optimal for N.

Interestingly, considering acyclic CP-nets, we mentioned earlier that an item o is optimal in N if and only if for every variable X, N contains the rule $o[\mathsf{pa}(X)]:o[X]>\overline{o[X]}$, therefore

$$P(\mathrm{opt}=o)=\prod_{X\in\mathscr{V}} p(X,o[\mathsf{pa}(X)]:o[X]>\overline{o[X]}).$$

This formula indicates that the probabilities of optimality can be encoded in a Bayesian Network associated to $\mathscr{N}$ [8]: let $BN(\mathscr{N})$ denote this network, it structure is the same oriented graph as that of $\mathscr{N}$, and, for every binary variable X and assignment $u\in\mathsf{pa}(X)$, the conditional probability table for X contains $p(x\,|\,u)=p(X,u:x>\bar{x})$. (For non binary variables, $p(x\,|\,u)$ would be the sum of the probabilities of the local orderings that have x at the top.) For instance, the probabilities of optimality of the PCP-net of Example 2 are encoded in the following Bayesian network:



In particular, computing the item that has the highest probability of being optimal is a #P-hard problem; If G is a tree, this item can be computed in linear time by using a bottom-up procedure [9].

Also, we can express the probability that a given value x for one variable $X\in\mathscr{V}$ appears in the optimal item as follows:

$$P(\mathrm{opt}[X]=x)=\sum_{u\in\underline{U}} p(X,u:x>\bar{x})\times P(\mathrm{opt}[U]=u)$$

where $\mathrm{opt}[U]=u$ denotes the event that the optimal item has values u for the variables in U. More generally, the probability that a partial assignment is optimal in terms of the probabilities of the rules of the PCP-net is:

$$P(\mathrm{opt}[U]=u)=\sum_{\substack{a\in\underline{\mathsf{asc}(U)}\\ a[U]=u}}\prod_{Y\in\mathsf{asc}(U)} p(Y,a[\mathsf{pa}(Y)]:a[Y]>\overline{a[Y]})$$

where $\mathsf{asc}(U)$ denotes the set of ascendants of the variables in U, including U. (More precisely, $\mathsf{asc}(U)$ is the smallest set that contains U and all the parents of each of its elements.) This equation does not give a practical mean of computing $P(\mathrm{opt}[U]=u)$ unless the number of parents and the height of the PCP-nets are bounded, since the size of $\underline{\mathsf{asc}(U)}$ is exponential in the size of $\mathsf{asc}(U)$.

2.1.2. Probability of dominance

Let, for any two items o,o', "$o\succ o'$" denote the set of compatible CP-nets N such that $o\succ_N o'$. Then $P(o\succ o')$ is, by definition, the sum of the probabilities $P(N)$ of the CP-nets N that are compatible with $\mathscr{N}$ and such that $o\succ_N o'$. [9] show that computing this probability is #P-complete, even when considering acyclic PCP-nets or polytrees.

3. Learning a PCP-net from probabilities of optimality

In many settings, like a recommender system, the system can record a list of items that can be assumed to have been optimal for some user at some point. It can be, for instance, a list of sold / rented items. Let $\mathscr{L}$ denote this list. Assuming that the users' preferences correspond to some PCP-net $\mathscr{N}$, the frequencies of the items in this list correspond to the probabilities of optimality of items in the corresponding Bayesian network. This suggests that this PCP-net can be induced from this list of sold items.

3.1. Off-line learning

Learning the parameters Let us assume that we know the structure of the hidden PCP-net, and that we want to estimate the probabilities in the tables. When the variables are binary, these probabilities are exactly the probabilities that appear in the tables of the Bayesian network that encodes the probabilities of optimality.

In particular, observing the probabilities of optimality can be sufficient to estimate the probabilities of the rules: for any binary variable $X \in \mathscr{V}$, for every $u \in \underline{\mathsf{pa}(X)}$, we have:

$$p(X,u:x\succ\bar{x}) = P(\mathrm{opt}[X]=x \mid \mathrm{opt}[U]=u) \sim |\{o\in\mathscr{L}, o[UX]=ux\}| / |\{o\in\mathscr{L}, o[U]=u\}|$$

when $\{o\in\mathscr{L}, o[U]=u\}$ is not empty, that is, when $P(\mathrm{opt}[U]=u)\neq 0$ in the hidden PCP-net.

More generally, we can use methods that have been used for Bayesian networks to learn these probabilities.

If $P(\mathrm{opt}[U]=u)=0$, we may still be able to estimate $p(X,u:x\succ\bar{x})$ from the probabilities of sub-optimal items, if we have a list of items that have been chosen by some users under some constraint, for instance a list of items sold during a period of a time where some options were not available. Assuming that the preferences of the user remain the same, the only effect of such constraint is to restrict the domain of some variables, but does not change the probabilities of other variables for the remaining combinations of values of the parents. Let $\mathscr{L}_{V=v}$ be a list of items optimal under the constraint $V=v$ for some $V\subseteq \mathsf{asc}(U)$ and $v\in\underline{V}$ such that $v[U]=u[V]$, and let "$\mathrm{opt}_{V=v}[U]=u$" denote the event that the item that is optimal among those that have values v for the variables in V, has values u for the variables in U, then

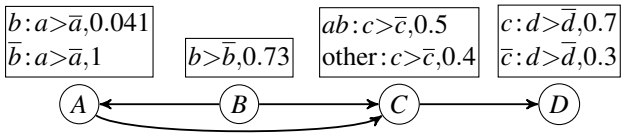
$$\begin{aligned} p(X,u:x\succ\bar{x}) &= P(\mathrm{opt}_{V=v}[X]=x \mid \mathrm{opt}_{V=v}[U]=u) \\ &\sim |\{o\in\mathscr{L}_{V=v}, o[UX]=ux\}| / |\{o\in\mathscr{L}_{V=v}, o[U]=u\}| \end{aligned}$$

For instance, consider a PCP-net that has the same structure as the PCP-net of Example 2. The probability of the rule $D,c:d\succ\bar{d}$ can be estimated as follows:

1. if $P(\mathrm{opt}[C]=c)\neq 0$:
   $p(D,c:d\succ\bar{d}) \sim |\{o\in\mathscr{L}, o[CD]=cd\}| / |\{o\in\mathscr{L}, o[C]=c\}|$;
2. if $P(\mathrm{opt}[C]=c)=0$, which is the case for instance if $p(a\succ\bar{a})=p(b\succ\bar{b})=1$ and $p(ab:c\succ\bar{c})=0$:
   $p(D,c:d\succ\bar{d}) \sim |\{o\in\mathscr{L}_{C=c}, o[D]=d\}| / |\mathscr{L}_{C=c}|$;
3. or, still if $P(\mathrm{opt}[C]=c)=0$, but $p(\bar{a}b:c\succ\bar{c})\neq 0$:
   $p(D,c:d\succ\bar{d}) \sim |\{o\in\mathscr{L}_{A=\bar{a}}, o[CD]=cd\}| / |\{o\in\mathscr{L}_{A=\bar{a}}, o[C]=c\}|$;

Equation 2 and 3 above give two different ways of computing $p(D,c:d>\overline{d})$ when the probability of having $C=c$ in a optimal item is zero, corresponding to two different observations: 2. corresponds to the observation of optimal items when $C=c$ is forced, and 3. to the observation of optimal items when $A=\overline{a}$ is forced.

Learning the structure Here again, methods used to learn Bayesian networks can be used. A major hurdle, however, is that several PCP-nets with different structures may give rise to the same probabilities of optimality: this is linked to the fact that the preferential dependencies encoded in PCP-net are oriented, whereas, in a Bayesian network, probabilistic dependencies are symetric. Consider for instance the Bayesian network that encodes the probabilities of optimality for the PCP-net of Example 2: it also encodes the probabilities of optimality of the PCP below, obtained from that of Example 2 by reversing the edge between A and B.



Therefore, methods for learning Bayesian networks will not completely identify a hidden PCP-net. However, quite a lot of information is obtained in this way. If a topological ordering of the otherwise unknown PCP-net is known, then the correct direction of each edge can be inferred, and the parameters of the hidden PCP net can be computed from the Bayesian network. Observe that there are natural situations in which such a general, total order might be known in advance. In particular, it is the case of interactive configuration, if the order of the variables to be configured is fixed (the system asks to choose a value for X_1, then one for X_2, etc.), then one can assume that the preferences will be expressed wrt this order, or at least, it makes sense to approximate these preferences on this order. Otherwise, the correct direction of the edges can be elicited, as described in the next section.

3.2. On-line learning

Structure elicitation Assuming that a Bayesian network encoding the probabilities of optimality has been computed, in an active learning setting some queries can lead to a quick identification of the correct orientation of the edges of the PCP-net. Assume for instance that the Bayesian network has an edge between variables X and Y: either the preferences over the domain of X depend on the value of Y, or the vise versa (but not both, since we assume an acyclic PCP net). In order to determine the orientation of the edge, one can submit to users the queries $x:y?\overline{y}$ and $\overline{x}:y?\overline{y}$, if the frequencies of the two possible answers to these queries converge to a common value over time, then y is preferentially independent of X. Otherwise, the preferences over the values of Y depend on the value of X, and X must be preferentially independent of Y.

Parameters update Finally, assume that a system has a current PCP net (maybe learnt from a list of past optimal items), and can now observe some users and their optimal items: it may be sensible to update the parameters (probabilities) of the PCP net according to what is being observed.

For instance, if the current PCP net corresponds to a group of past users, and a new user connects to the system, her preferences may not be exactly that of the group, and we may want to modify the parameters of the PCP net so that it incorporates the preferences of this particular user. Or it may be the case that the PCP net represents probabilities of preferences at a given time, or in a given location, and that these probabilities need to be updated in a new context.

Since the probabilities that we want to update directly correspond to the probabilities of some items being optimal, the observations made by the system must be about optimality. Every time our system is presented an optimal item o, it will update its parameters, that is, the probabilities of its current PCP-net $\mathscr{N}$ as follows:

For every observed optimal item o do: for every $X \in \mathscr{V}$ do:

1. let $u = o[\mathsf{pa}(U)]$;
2. let $X_o = 1$ if $o[X] = x$, 0 otherwise;
3. $p(X,u:x>\bar{x})$ += $\eta_t (X_o - p(X,u:x>\bar{x}))$.

Note that we only update the probabilities of some of the rules (step 1.): the rules that make the given item optimal. The update rule is common in such a stochastic learning setting (see e.g. [3]). The parameter η_t is the *learning rate*, it may vary over time, generally decrease in order to ensure convergence: convergence is guaranteed if $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$. In our experiments we took $\eta_t = 1/k(t,X,u)$ where $k(t,X,u)$ is the number of times the rule corresponding to (X,u) has been updated so far (making the learning rate a function of the pair (X,u)). In this case, at any time, $p(X,u:x>\bar{x})$ is just the frequency with which optimal items o with $o[U] = u$ and $o[X] = x$ have been encountered so far.

4. Experiments

We ran some experiments to evaluate the update rule that we proposed for an on-line learning setting. We assume a given acyclic structure over a given set of n variables. We have some hidden PCP-net $\mathscr{N}^*$ with this structure, and start from an initial PCP-net with the same structure, where the probabilities of all rules are $1/2$. At each step, we update some rules of the current PCP-net $\mathscr{N}$.

We observed how the distance between the target PCP-net $\mathscr{N}^*$ and the current one $\mathscr{N}$ evolved. As measures of distances we used:

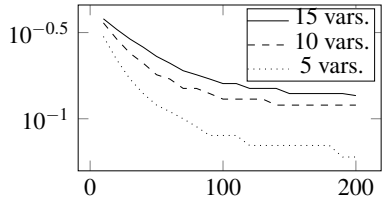
- the sum of the squared differences of the parameters

$$d_2(\mathscr{N},\mathscr{N}^*) = \sum_{(X,u)} \left(p_{\mathscr{N}}(X,u:x>\bar{x}) - p_{\mathscr{N}^*}(X,u:x>\bar{x}) \right)^2;$$

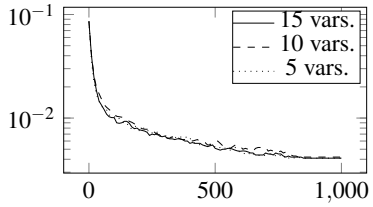
- the Kullback-Leibler divergence, or relative entropy, of the two probability distributions of optimality defined by $\mathscr{N}$ and $\mathscr{N}^*$:

$$d_{KL}(\mathscr{N} \| \mathscr{N}^*) = \sum_{o \in \underline{\mathscr{V}}} \log\left(P_{\mathscr{N}}(\mathrm{opt}=o) / P_{\mathscr{N}^*}(\mathrm{opt}=o) \right) \times P_{\mathscr{N}}(\mathrm{opt}=o).$$

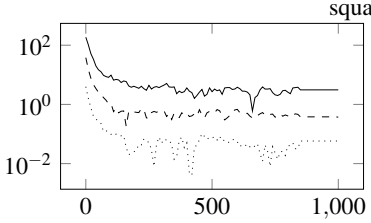
In fact, we computed an estimate of this distance by sampling the set of items, assuming a uniform distribution.



First protocol:Sum of squared diff. of param.



Second protocol:Sum of squared diff. of param.



Second protocol – KL divergence

Figure 1. Plots showing the evolution, as the number of observations of optimal outcomes grows (x-axis), of a measure of the distance between the target PCP-net and the learnt one (y-axis)

We have experimented with two protocols to generate optimal items at each time step.

First protocol The idea is to simulate an interactive setting, in which, for some pair of items o_1 and o_2, we observe for a while which is most frequently optimal, and update our current PCP-net if it does not give the same result. More precisely, for each new period we generate two items as follows: we generate two CP-nets N_1 and N_2 according to the distribution defined by our current hypothesis $\mathscr{N}$ (this is achieved by choosing, for each combination (X,u), the rule $X,u:x>\bar{x}$) according to the current probability $p(X,u:x>\bar{x})$; let o_1 and o_2 be the respective optimal items of N_1 and N_2. Generating the two items in this manner will favor rules that are more probable so far. We can compute the probabilities that o_1 and o_2 are optimal according to our current PCP-net $\mathscr{N}$: let $p_i = P_{\mathscr{N}}(\text{opt} = o_i)$. We then "observe" which of o_1 and o_2 is most frequently optimal according to the hidden PCP-net $\mathscr{N}^*$: in fact, we compute $p_i^* = P_{\mathscr{N}^*}(\text{opt} = o_i)$. Eventually, if $p_1 > p_2$ whereas $p_2^* > p_1^*$, we update $\mathscr{N}$ so as to increase the probability that o_2 is optimal: we use the update algorithm above with $o = o_2$.

Second protocol Here we just simulate the update of our PCP-net after each newly observed chosen item, assuming that it is optimal for some user: we increase the probabilities of the rules that make this item optimal. Therefore, at each time step, we generate a "user" / PCP-net N according to the target distribution represented by the target PCP-net $\mathscr{N}^*$, compute its optimal item o, and run the update algorithm with o.

Results We have run 500 trials with random target PCP-nets with 5, 10 and 15 variables. For each trial, we generated a target PCP-net, ran our experimental protocol and learning algorithm, and measured the distance between the learnt PCP-net and the target one, every 10 observations of an optimal item. The plots in Figure 1 depict the evolution of this distance: each point is an average over the 500 trials for each number of variables.

As can be noticed, a good approximation of the target PCP-net is reached after around 70 observations.

5. Conclusion

We have described in this paper how it is possible to learn a probabilistic representation of the preferences of a group of users over a combinatorial domain, or how we can fine-tune such preferences to fit more precisely one particular user. Since CP-nets in general are good at finding most preferred items, our learning method supposes that the learner can be supplied with a list of past most preferred items: we showed how a probabilistic CP-net can be learnt from such information.

References

[1] D. Bigot, H. Fargier, J. Mengin, and B. Zanuttini. Using and learning gai-decompositions for representing ordinal rankings. In *Proc. ECAI workshop on Preference Learning*, pages 5–10, 2012.

[2] R. Booth, Y. Chevaleyre, J. Lang, J. Mengin, and C. Sombattheera. Learning conditionally lexicographic preference relations. In *Proc. ECAI 2010*.

[3] L. Bottou. Stochastic learning. In O. Bousquet, U. von Luxburg, and G. Rätsch, editors, *Advanced Lectures on Machine Learning*, LNCS 3176, pages 146–168. Springer, 2004.

[4] C. Boutilier, F. Bacchus, and R. I. Brafman. UCP-networks: A directed graphical representation of conditional utilities. In *Proc. UAI 2001*, pages 56–64. Morgan Kaufmann.

[5] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artificial Intelligence Research*, 21:135–191, 2004.

[6] R. I. Brafman and C. Domshlak. Introducing variable importance tradeoffs into CP-nets. In *Proc UAI 2002*, pages 69–76.

[7] C. Cornelio. Dynamic and probabilistic cp-nets. Master's thesis, University of Padua, 2012.

[8] C. Cornelio, J. Goldsmith, N. Mattei, F. Rossi, and K. B. Venable. Updates and uncertainty in CP-nets. In *Proc. Australasian Joint Conf. on Advances in Art. Intell. 2013*, LNCS 8272 , pages 301–312. Springer.

[9] D. D. Bigot, H. Fargier, J. Mengin, and B. Zanuttini. Probabilistic conditional preference networks. In A. Nicholson and P. Smyth, editors, *Proc. UAI 2013.*

[10] Y. Dimopoulos, L. Michael, and F. Athienitou. Ceteris paribus preference elicitation with predictive guarantees. In *Proc. IJCAI 2009*.

[11] J. Fürnkranz and H. Hüllermeier, editors. *Preference learning*. Springer, 2011.

[12] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. KDD 2002*, pages 133–142..

[13] F. Koriche and B. Zanuttini. Learning conditional preference networks with queries. In *Proc. IJCAI 2009*.

[14] J. Lang and J. Mengin. The complexity of learning separable ceteris paribus preferences. In *Proc. IJCAI 2009*.

[15] D. Mailharro. A classification and constraint-based framework for configuration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 12(4):383–397, 1998.

[16] S. S. de Amo, M. Bueno, G. Alves, and N. Silva. CPrefMiner: An algorithm for mining user contextual preferences based on bayesian networks. In *Proc. ICTAI 2012*, vol. 1, pages 114–121.

[17] P. Viappiani, B. Faltings, and P. Pu. Preference-based search using example-critiquing with suggestions. *J. Artificial Intelligence Research*, 27:465–503, 2006.

[18] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In W. Kraaij, A. P. de Vries, C. L. A. Clarke, N. Fuhr, and N. Kando, editors, *Proc. 30th ACM SIGIR Conf. on Information Retrieval 2007*, pages 391–398.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-91

A Logic of Part and Whole for Buffered Geometries

Heshan DU [a] and Natasha ALECHINA [a]
[a] *University of Nottingham, UK*

Abstract. We propose a new qualitative spatial logic for reasoning about part-whole relations between geometries (sets of points) represented in different geospatial datasets, in particular crowd-sourced datasets. Since geometries in crowd-sourced data can be less inaccurate or precise, we buffer geometries by a margin of error or level of tolerance σ, and define part-whole relation for buffered geometries. The relations between geometries considered in the logic are: buffered part of (BPT), Near and Far. We provide a sound and complete axiomatisation of the logic with respect to metric models and show that its satisfiability problem is NP-complete.

1. MOTIVATION

This work is motivated by our previous work [3] on integrating authoritative geospatial information and crowd-sourced or volunteered geospatial information. Geometry representations of the same location or place in different datasets are usually not exactly the same. Objects are also sometimes represented at different levels of granularity. For example, consider geometries of objects in Nottingham city centre given by the Ordnance Survey of Great Britain (OSGB) [7] and by the OpenStreetMap (OSM) [6] in Figure 1. The position and shape of the Prezzo Ristorante are represented differently in OSGB (dotted) and OSM (solid) (Figure 1.a). The Victoria Shopping Centre is represented as a whole in OSM (Figure 1.b), and as several shops in OSGB (Figure 1.c).

In order to integrate the datasets, we need to determine which objects are the same and sometimes (as in the example of Victoria Shopping Centre) which objects in one dataset are parts of objects in another. One way to produce such matches is to use locations and geometries of objects, although of course we also use any lexical labels associ-

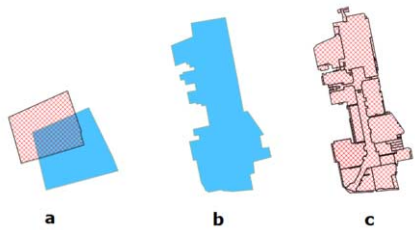


Figure 1. a. Prezzo Ristorante; **b.** Victoria Shopping Centre in OSM; **c.** Victoria Shopping Centre in OSGB

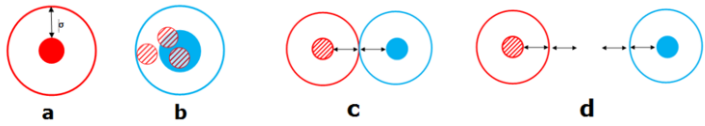


Figure 2. **a.** a buffer; **b.** three dashed circles are buffered part of (BPT) the solid circle; **c.** NEAR; **d.** FAR

ated with the objects, such as names of restaurants etc. The generated matches are seen as assumptions, and are retractable if found incorrect. We check correctness of matches by checking their logical consistency. Some of the checks use ontology reasoning (if an object is classified as a restaurant in one dataset and as a bank in another, together with an axiom stating that the concepts of Restaurant and Bank are disjoint, a contradiction can be derived). Other checks are performed using spatial reasoning. In [4], we proposed a spatial logic LNF that contains relations of being buffered equal (BEQ), Near and Far to validate 'sameAs' matches of objects: if it is conjectured that a_1 is 'sameAs' b_1 and that a_2 is 'sameAs' b_2, then a contradiction can be derived if $NEAR(a_1, a_2)$ and $FAR(b_1, b_2)$. However, LNF is not appropriate for verifying 'partOf' matches. In this paper, we are proposing a logic where we can formalise for example the following argument: if b and c are near, c is part of d, then b cannot be part of a which is far from d. The main concepts of this logic, which we call a Logic of ParT and whole for Buffered geometries (LBPT), are explained in the next section. We also compare it to existing spatial logics.

2. BPT, NEAR, FAR

For the application described in the previous section, we found it difficult to use formalisms such as RCC and other topology or mereology theories [1], since they presuppose accurate geometries or regions with sharp boundaries. Unlike existing models for spatial relations between indeterminate regions or objects with broad boundaries based on rough set theory [8], such as [2] and [9], we could not define a certain inner region, because the same location can be represented using two disconnected polygons from authoritative and crowd-sourced geospatial datasets respectively, which requires that the whole region within the buffer [5] of a geometry is uncertain. We did not adopt probabilistic or fuzzy approaches, such as [12] and [10], because we did not have a good way to define a proper probability function or a membership function for a fuzzy set. The first logic we designed for debugging geometry matches, LNF [4], has the 'buffered equal' relation as a basic relation, which turns out to be less useful when the data is represented at different levels of abstraction (such as a shopping centre in one set and a collection of shops in another). In [4], we gave a complete and sound axiomatisation for LNF, but only with respect to geometries consisting of a single point. In this paper, we start with the 'buffered part of' (BPT) relation as the basic relation, and interpret geometries as sets of points.

As shown in Figure 2.a, by buffering the solid circle c by σ, where σ indicates the margin of error or level of tolerance, we obtain a larger circle, denoted as $buffer(c, \sigma)$, where every point is within σ distance from c. For a geometry c which is possibly represented inaccurately within the margin of error σ, the actual accurate representation is assumed to be somewhere within $buffer(c, \sigma)$. A geometry g is buffered part of a geometry h, if g is within $buffer(h, \sigma)$ (Figure 2.b).

We also have NEAR and FAR relations in the logic LBPT. They formalise concepts of being 'possibly connected' (given a possible displacement by σ) and 'definitely disconnected' (even if displaced by σ) respectively. Two geometries are possibly connected iff their σ buffers are connected. Figure 2.c and Figure 2.d show the boundary case of being NEAR, where $distance(g,h) = 2\sigma$ (their buffers are externally connected) and the case where two geometries are far apart and cannot possibly correspond to connected objects respectively.

3. SYNTAX, SEMANTICS AND AXIOMS OF LBPT

The language $L(LBPT)$ contains a set of individual names, three binary predicates $NEAR$, FAR and BPT, and logical connectives, $\neg, \wedge, \vee, \rightarrow$.

Applying predicate letters to individual names yields atomic formulas, e.g. $BPT(a,b)$. Every atomic formula is a well-formed formulas (wffs). If α and β are wffs, then $\neg\alpha$, $\alpha \wedge \beta$, $\alpha \vee \beta$, $\alpha \rightarrow \beta$ are wffs.

We interpret the logic over models which are based on a metric space (similar to other spatial logics, such as [13] and [1], and also similar to [11] but for a different logical language).

Definition 1 (Metric Space) *A* metric space *is a pair (Δ, d), where Δ is a set and d is a metric on Δ, i.e., a function $d : \Delta \times \Delta \longrightarrow \mathbb{R}_{\geq 0}$ such that for any $x, y, z \in \Delta$, the following holds:*

1. $d(x,y) = 0$ *iff* $x = y$;
2. $d(x,y) = d(y,x)$;
3. $d(x,z) \leq d(x,y) + d(y,z)$.

Definition 2 (Metric Model) *A metric model M is a tuple (Δ, d, I, σ), where (Δ, d) is a metric space, I is an interpretation function which maps each constant to a set of elements in Δ, and $\sigma \in \mathbb{R}_{>0}$ is the margin of error. The notion of $M \models \phi$ (ϕ is true in model M) is defined as follows:*

$M \models BPT(a,b)$ *iff* $\forall p_a \in I(a) \exists p_b \in I(b) : d(p_a, p_b) \in [0, \sigma]$;
$M \models NEAR(a,b)$ *iff* $\exists p_a \in I(a) \exists p_b \in I(b) : d(p_a, p_b) \in [0, 2\sigma]$;
$M \models FAR(a,b)$ *iff* $\forall p_a \in I(a) \forall p_b \in I(b) : d(p_a, p_b) \in (4\sigma, +\infty)$;
$M \models \neg\alpha$ *iff* $M \not\models \alpha$;
$M \models \alpha \wedge \beta$ *iff* $M \models \alpha$ *and* $M \models \beta$;
$M \models \alpha \vee \beta$ *iff* $M \models \alpha$ *or* $M \models \beta$;
$M \models \alpha \rightarrow \beta$ *iff* $M \not\models \alpha$ *or* $M \models \beta$

where a, b are individual names, α, β are wffs.

A formula α is valid ($\models \alpha$) if for every metric model M, $M \models \alpha$. The logic LBPT is the set of all valid formulas of $L(LBPT)$.

The following calculus (that we will also refer to as LBPT) will be shown sound and complete for LBPT:

Axiom 0 All tautologies of classical propositional logic;
Axiom 1 $BPT(a,a)$;
Axiom 2 $NEAR(a,b) \rightarrow NEAR(b,a)$;

Axiom 3 $FAR(a,b) \rightarrow FAR(b,a)$;
Axiom 4 $BPT(a,b) \wedge BPT(b,c) \rightarrow NEAR(c,a)$;
Axiom 5 $BPT(b,a) \wedge BPT(b,c) \rightarrow NEAR(c,a)$;
Axiom 6 $BPT(b,a) \wedge NEAR(b,c) \wedge BPT(c,d) \rightarrow \neg FAR(d,a)$;
Axiom 7 $NEAR(a,b) \wedge BPT(b,c) \wedge BPT(c,d) \rightarrow \neg FAR(d,a)$;
MP Modus ponens: $\phi,\ \phi \rightarrow \psi \vdash \psi$.

The notion of derivability $\Gamma \vdash \phi$ in LBPT is standard. A formula ϕ is LBPT-derivable if $\vdash \phi$. A set Γ is (LBPT) inconsistent if for some formula ϕ it derives both ϕ and $\neg\phi$.

4. SOUNDNESS AND COMPLETENESS OF LBPT

In this section we prove that LBPT is sound and complete with respect to metric models, namely that $\vdash \phi \ \Leftrightarrow \models \phi$. Proofs of some lemmas are omitted due to lack of space. Detailed proofs can be found here: `www.cs.nott.ac.uk/~hxd/report/lbpt.pdf`.

Theorem 1 (Soundness) *Every LBPT derivable formula is valid:* $\vdash \phi \Rightarrow \models \phi$.

Proof. The proof is by an easy induction on the length of the derivation of ϕ. Axioms 1-7 are valid (by the truth definition of *BPT*, *NEAR* and *FAR*) and modus ponens preserves validity. QED.

In the rest of this section, we prove completeness. We will actually prove that given a finite consistent set of formulas, we can build a satisfying model for it. This shows that $\nvdash \phi \Rightarrow \not\models \phi$ and by contraposition we get completeness. The completeness proof is more involved than that for LNF with respect to point geometries [4].

Definition 3 (MCS) *A set of formulas* Γ *in the language* $L(LBPT)$ *is maximal consistent, if* Γ *is consistent, and any set of LBPT formulas over the same set of individual names properly containing* Γ *is inconsistent. If* Γ *is a maximal consistent set of formulas, then we call it an MCS.*

Lemma 1 (Lindenbaum's Lemma) *If* Σ *is a consistent set of formulas in the language* $L(LBPT)$*, then there is an MCS* Σ^+ *over the same set of individual names such that* $\Sigma \subseteq \Sigma^+$.

Let $\phi_0, \phi_1, \phi_2, \ldots$ be an enumeration of LBPT formulas over the same set of individual names as that in Σ. Σ^+ can be defined as follows:

- $\Sigma_0 = \Sigma$;
- $\Sigma_{n+1} = \Sigma_n \cup \{\phi_n\}$, if it is consistent, otherwise, $\Sigma_{n+1} = \Sigma_n \cup \{\neg\phi_n\}$;
- $\Sigma^+ = \bigcup_{n \geq 0} \Sigma_n$.

Given a consistent set of formulas Σ, we construct a metric model satisfying a maximal consistent set Σ^+ containing Σ, following the steps below.

Step 1 We equivalently transform Σ^+ to $B(\Sigma^+)$, a set of basic quantified formulas.
Step 2 We construct a set of distance constraints $D(\Sigma^+)$ from $B(\Sigma^+)$, such that any metric space satisfying $D(\Sigma^+)$ can be extended to a model of $B(\Sigma^+)$.

Step 3 We show that if $D(\Sigma^+)$ is path-consistent, then there is a metric space (Δ, d) satisfying $D(\Sigma^+)$.

Step 4 We show that $D(\Sigma^+)$ is path-consistent, if Σ^+ is consistent.

Since Σ^+ is consistent, then there is a metric space that can be extended to a metric model satisfying $B(\Sigma^+)$, thus, Σ^+, thus, Σ.

In **Step 1**, we equivalently transform Σ^+ to a set of basic quantified formulas defined as follows.

Definition 4 (Basic Quantified Formula) *Observe that atomic LBPT formulas are equi-satisfiable with first order quantified formulas corresponding to their truth conditions in Definition 2:*

- $BPT(a,b)$ *and the formula* $\forall p_a \in a \exists p_b \in b : d(p_a,p_b) \in [0,\sigma]$ *are equi-satisfiable;*
- $NEAR(a,b)$ *and the formula* $\exists p_a \in a \exists p_b \in b : d(p_a,p_b) \in [0,2\sigma]$ *are equi-satisfiable;*
- $FAR(a,b)$ *and the formula* $\forall p_a \in a \forall p_b \in b : d(p_a,p_b) \in (4\sigma,\infty)$ *are equi-satisfiable.*

We refer to these first order quantified formulas as basic quantified formulas, and use the following abbreviations for them, where g is a non-negative interval.

- $\forall(a,b,g) \equiv (\forall p_a \in a \forall p_b \in b : d(p_a,p_b) \in g)$;
- $\exists(a,b,g) \equiv (\exists p_a \in a \exists p_b \in b : d(p_a,p_b) \in g)$;
- $\chi(a,b,g) \equiv (\forall p_a \in a \exists p_b \in b : d(p_a,p_b) \in g)$;
- $\xi(a,b,g) \equiv (\exists p_a \in a \forall p_b \in b : d(p_a,p_b) \in g)$.

Lemma 2 *For any MCS* Σ^+ *and any pair of individual names* a,b *occurring in* Σ*, exactly one of the following cases holds:*

C1 $BPT(a,b) \wedge BPT(b,a) \in \Sigma^+$;
C2 $BPT(a,b) \wedge \neg BPT(b,a) \in \Sigma^+$;
C3 $\neg BPT(a,b) \wedge BPT(b,a) \in \Sigma^+$;
C4 $\neg BPT(a,b) \wedge \neg BPT(b,a) \wedge NEAR(a,b) \in \Sigma^+$;
C5 $\neg NEAR(a,b) \wedge \neg FAR(a,b) \in \Sigma^+$;
C6 $FAR(a,b) \in \Sigma^+$.

Definition 5 (B(Σ^+)) *Given an MCS* Σ^+*, a corresponding set of basic quantified formulas* $B(\Sigma^+)$ *is constructed as follows. For every pair of individual names* a,b*, we translate the LBPT formulas to basic quantified formulas:*

- $translate(BPT(a,b) \wedge BPT(b,a)) = \{\chi(a,b,[0,\sigma]), \chi(b,a,[0,\sigma])\}$;
- $translate(BPT(a,b) \wedge \neg BPT(b,a)) = \{\chi(a,b,[0,\sigma]), \xi(b,a,(\sigma,\infty))\}$;
- $translate(\neg BPT(a,b) \wedge BPT(b,a)) = \{\xi(a,b,(\sigma,\infty)), \chi(b,a,[0,\sigma])\}$;
- $translate(\neg BPT(a,b) \wedge \neg BPT(b,a) \wedge NEAR(a,b)) =$
  $\{\xi(a,b,(\sigma,\infty)), \xi(b,a,(\sigma,\infty)), \exists(a,b,[0,2\sigma]), \exists(b,a,[0,2\sigma])\}$;
- $translate(\neg NEAR(a,b) \wedge \neg FAR(a,b)) =$
  $\{\forall(a,b,(2\sigma,\infty)), \forall(b,a,(2\sigma,\infty)), \exists(a,b,[0,4\sigma]), \exists(b,a,[0,4\sigma])\}$;
- $translate(FAR(a,b)) = \{\forall(a,b,(4\sigma,\infty)), \forall(b,a,(4\sigma,\infty))\}$.

Let $names(\Sigma)$ *be the set of individual names occurring in* Σ*. Then,*

$$B(\Sigma^+) = \bigcup_{a \in names(\Sigma), b \in names(\Sigma)} translate(case(a,b))$$

where case(a,b) returns the LBPT formula in the case of a,b specified in Lemma 2.

By the construction above, $B(\Sigma^+)$ contains the same set of individual names as Σ, and any metric model satisfying $B(\Sigma^+)$ satisfies Σ^+.

In **Step 2**, for a set of basic quantified formulas $B(\Sigma^+)$, we construct a set of distance constraints $D(\Sigma^+)$, and then show that if there is a metric space satisfying $D(\Sigma^+)$, then it can be extended to a model of $B(\Sigma^+)$ (hence Σ^+).

Next we turn to producing enough 'points' to populate geometries corresponding to individual names. The next definition specifies the cardinality of the set *points*(a) (points assigned to an individual name a).

Definition 6 ($num_{B(\Sigma^+)}(a)$**)** *Let names(Σ) be the set of individual names occurring in Σ, $B(\Sigma^+)$ is a corresponding set of basic quantified formulas of Σ^+, an MCS of Σ. For any individual name $a \in$ names(Σ),*

$$num_{B(\Sigma^+)}(\exists a) = |\{b \in names(\Sigma) \mid \exists g : \exists(a,b,g) \in B(\Sigma^+)\}|$$
$$num_{B(\Sigma_+)}(\xi a) = |\{b \in names(\Sigma) \mid \exists g : \xi(a,b,g) \in B(\Sigma^+)\}|$$
$$num_{B(\Sigma^+)}(\chi a) = |\{b \in names(\Sigma) \mid \exists g : \chi(b,a,g) \in B(\Sigma^+)\}|$$

Then $num_{B(\Sigma^+)}(a) = \max(1, num_{B(\Sigma^+)}(\exists a) + num_{B(\Sigma^+)}(\xi a) + num_{B(\Sigma^+)}(\chi a))$.

We omit subscript $B(\Sigma^+)$ for readability when it is clear from context.

Definition 7 (Witness for a formula) *A witness for a formula $\exists(a,b,g)$ is a pair of constants $p_a \in a$, $p_b \in b$ such that $d(p_a,p_b) \in g$. A witness for a formula $\xi(a,b,g)$ or $\chi(b,a,g)$ is a constant $p_a \in a$, such that $d(p_a,p_b) \in g$, for any constant $p_b \in b$. A constant is clean for a formula, if it is not a witness for any other formula.*

Definition 8 (D(Σ^+)) *Let $B(\Sigma^+)$ be the set of basic quantified formulas of an MCS Σ^+. To every individual name a in Σ, we assign a fixed set of new constants, points$(a) = \{p_a^1, \ldots, p_a^n\}$, where $n = num(a)$. We construct a set of distance constraints $D(\Sigma^+)$ as follows, by iterating through basic quantified formulas in $B(\Sigma^+)$ and eliminating quantifiers on new constants. Initially, $D(\Sigma^+) = \{\}$. For every individual name a in Σ, for every constant $p_a \in points(a)$, we add $d(p_a,p_a) \in \{0\}$ to $D(\Sigma^+)$. Then $\chi(a,a,\{0\})$ always holds. For every pair of different individual names a,b, if*

- *$\exists(a,b,g) \in B(\Sigma^+)$, then we take clean constants $p_a \in points(a)$, $p_b \in points(b)$, and add $d(p_a,p_b) = d(p_b,p_a) \in g$ to $D(\Sigma^+)$ (p_a, p_b become the witness for $\exists(a,b,g)$);*
- *$\xi(a,b,g) \in B(\Sigma^+)$, then we take a clean constant $p_a \in points(a)$, for every $p_b \in points(b)$, we add $d(p_a,p_b) = d(p_b,p_a) \in g$ to $D(\Sigma^+)$;*
- *$\xi(b,a,g) \in B(\Sigma^+)$, then we take a clean constant $p_b \in points(b)$, for every $p_a \in points(a)$, we add $d(p_a,p_b) = d(p_b,p_a) \in g$ to $D(\Sigma^+)$;*
- *$\chi(a,b,g) \in B(\Sigma^+)$, then we take a clean constant $p_b \in points(b)$, for every $p_a \in points(a)$, we add $d(p_a,p_b) = d(p_b,p_a) \in g$ to $D(\Sigma^+)$;*
- *$\chi(b,a,g) \in B(\Sigma^+)$, then we take a clean constant $p_a \in points(a)$, for every $p_b \in points(b)$, we add $d(p_a,p_b) = d(p_b,p_a) \in g$ to $D(\Sigma^+)$;*
- *$\forall(a,b,g) \in B(\Sigma^+)$, then for every pair of constants $p_a \in points(a)$, $p_b \in points(b)$, we add $d(p_a,p_b) = d(p_b,p_a) \in g$ to $D(\Sigma^+)$.*

For every pair of different constants p,q involved in $D(\Sigma^+)$, we add $d(p,q) = d(q,p) \in [0,\infty)$ to $D(\Sigma^+)$, then repeatedly replace $d(p,q) = d(q,p) \in g_1$ and $d(p,q) = d(q,p) \in g_2$ with $d(p,q) = d(q,p) \in (g_1 \cap g_2)$, until there is one distance range for each pair of p,q.

Lemma 3 *Any metric space satisfying $D(\Sigma^+)$ can be extended to a model of $B(\Sigma^+)$.*

Proof.(sketch) Suppose S is a metric space satisfying $D(\Sigma^+)$. We extend S to a model M by interpreting every a occurring in Σ as $points(a)$, as specified in Definition 8. We can prove that for any individual name a, $points(a)$ covers all the clean constants needed for constructing $D(\Sigma^+)$. Then by Definition 8, every basic quantified formula has a witness. Therefore M is a model of $B(\Sigma^+)$. QED.

$D(\Sigma^+)$ and $B(\Sigma^+)$ are not equi-satisfiable because of the way we assign witnesses for χ formulas, but we will show that if $B(\Sigma^+)$ is consistent then $D(\Sigma^+)$ can be satisfied in a metric space. The following definitions are essential for **Step 3** and **Step 4**.

Definition 9 (Composition) *For non-negative numbers d_1, d_2, the composition $\{d_1\} \circ \{d_2\} = [|d_1 - d_2|, d_1 + d_2]$ [1]. For non-negative intervals g_1, g_2, their composition $g_1 \circ g_2 = \bigcup_{d_1 \in g_1, d_2 \in g_2} \{d_1\} \circ \{d_2\}$.*

Definition 10 (Path Consistency) *Given a set of distance constraints D, for every pair of constants a,b, their distance range is strengthened by enforcing path-consistency as follows until a fixed point is reached:*

$$\forall c : g(a,b) \leftarrow g(a,b) \cap (g(a,c) \circ g(c,b))$$

where c is a constant different from a,b, $g(a,b)$ denotes the distance range for a,b (i.e. $d(a,b) \in g(a,b)$). If at the fixed point, for every pair of constants a,b, there exists a valid value for their distance, this is, $g(a,b) \neq \emptyset$, then D is path-consistent.

Definition 11 (Primitive, Composite, Definable Intervals) *Let h be a non-negative interval. h is primitive, if h is one of $[0,\sigma]$, (σ,∞), $[0,2\sigma]$, $(2\sigma,\infty)$, $(2\sigma,4\sigma]$, $(4\sigma,\infty)$, $[0,\infty)$. h is composite, if it can be composed using at least two primitive intervals. h is definable, if it is primitive or composite.*

It is easy to show that if an interval occurs in $D(\Sigma^+)$, then it is an identity interval ($\{0\}$) or a primitive interval.

Definition 12 ($DS(\Sigma^+)$) *We define the set of distance constraints which appear in the process of enforcing path-consistency on $D(\Sigma^+)$, denoted as $DS(\Sigma^+)$, as follows:*

- *Any distance constraint in $D(\Sigma^+)$ is in $DS(\Sigma^+)$;*
- *If distance constraints $d(a,b) \in h$ and $d(b,c) \in g$ are in $DS(\Sigma^+)$, then $d(a,c) \in h \circ g$ is in $DS(\Sigma^+)$;*
- *If distance constraints $d(a,b) \in h$ and $d(a,b) \in g$ are in $DS(\Sigma^+)$, then $d(a,b) \in h \cap g$ is in $DS(\Sigma^+)$*

where a,b,c are constants in $D(\Sigma^+)$.

[1] Based on $d(x,z) \leq d(x,y) + d(y,z)$ (Property 3 of Definition 1).

For a distance constraint $d(a,b) \in h$ in $DS(\Sigma^+)$, we proved that h is a non-negative interval; h is either right-infinite or right-closed; if $lower(h) \neq 0$, then h is left-open.

We are now going to characterise all possible distance constraints occurring in $DS(\Sigma^+)$. Eventually, we will show that all those distance constraints are left and right definable in the sense given below. For an interval h of the form (l,u), $[l,u)$, $(l,u]$ or $[l,u]$, we call l the lower bound of h, represented as $lower(h)$, and u the upper bound of h, represented as $upper(h)$. We allow $lower(h)$ or $upper(h)$ to be ∞. For the lower or upper bound of an interval h, we use $^-$ or $^+$ to denote that h is open or closed respectively.

Definition 13 (Left-Definable) *A distance constraint $d(c_1,c_n) \in h$ $(n > 1)$ is left-definable, iff there exists a sequence of distance constraints $d(c_i,c_{i+1}) \in h_i$ $(0 < i < n)$ in $D(\Sigma^+)$, such that, for $h' = h_1 \circ ... \circ h_{n-1}$, the following holds:*

1. *If h is left-open, then h' is left-open, $h \subseteq h'$, and $lower^-(h') = lower^-(h)$;*
2. *If h is left-closed, then h' is left-closed, $h \subseteq h'$, and $lower^+(h') = lower^+(h)$.*

Definition 14 (Right-Definable) *A distance constraint $d(c_1,c_n) \in h$ $(n > 1)$ is right-definable, iff there exists a sequence of distance constraints $d(c_i,c_{i+1}) \in h_i$ $(0 < i < n)$ in $D(\Sigma^+)$, such that, for $h' = h_1 \circ ... \circ h_{n-1}$, the following holds:*

1. *If h is right-open, then h' is right-open, $h \subseteq h'$, and $upper^-(h') = upper^-(h)$;*
2. *If h is right-closed, then h' is right-closed, $h \subseteq h'$, and $upper^+(h') = upper^+(h)$.*

Lemma 4 *If a distance constraint $d(a,b) \in h$ is in $DS(\Sigma^+)$, then it is left-definable and right-definable.*

Lemma 4 can be proved by an induction on the number of operations (intersection or composition) applied, to obtain $d(a,b) \in h$ from $D(\Sigma^+)$.

In **Step 3**, following the same way as described in [4], we can construct a metric space satisfying all the constraints in $D(\Sigma^+)$. The main lemmas proved are stated below.

Lemma 5 *Let t be the number of constants in $D(\Sigma^+)$. Enforcing path-consistency on $D(\Sigma^+)$, a fixed point can be reached in $O(t^3)$.*

For any interval h occurring in $D(\Sigma^+)$, $h \subseteq [0,\infty)$. In the worst case, $[0,\infty)$ can be strengthened at most $4t$ times (first strengthen it to $[0,u]$, $u \leq 4\sigma(t-1)$, then strengthen it by σ each time). For t constants, there are $O(t^2)$ distance constraints in $D(\Sigma^+)$. Therefore, the total time of strengthening all the distance constraints is $O(t^3)$.

Lemma 6 *Let t be the number of constants in $D(\Sigma^+)$, $D^f(\Sigma^+)$ be a fixed point of enforcing path consistency on $D(\Sigma^+)$. If $D(\Sigma^+)$ is path-consistent, $D_s(\Sigma^+)$ is obtained from $D^f(\Sigma^+)$ by replacing every right-infinite interval with $\{5t\sigma\}$, every right-closed interval h with $\{upper(h)\}$, then $D_s(\Sigma^+)$is path-consistent.*

Any interval referred in $D^f(\Sigma^+)$ is either right-infinite or right-closed.

Lemma 7 *Let Σ^+ be an MCS. If $D(\Sigma^+)$ is path-consistent, then there is a metric space (Δ,d) such that all the constraints in $D(\Sigma^+)$ are satisfied by d.*

In **Step 4**, we will prove $D(\Sigma^+)$ is path-consistent by contradiction.

Lemma 8 *Let Σ^+ be an MCS. Then its set of distance constraints $D(\Sigma^+)$ is path-consistent.*

Proof.(sketch) Suppose $D(\Sigma^+)$ is not path-consistent. By Definitions 10 and 12, $d(p,q) \in \emptyset$ is in $DS(\Sigma^+)$, for some constants p,q. It is easy to show that for any distance range g occurring in $D(\Sigma^+)$, $g \neq \emptyset$. By Definitions 12, 9, and intersection rules, the last operation to obtain the first $\emptyset$ interval is intersection. By Definition 12, there exist $d(p,q) \in h$ and $d(p,q) \in g$ in $DS(\Sigma^+)$, $h \neq \emptyset$, $g \neq \emptyset$, and $h \cap g = \emptyset$. h,g are non-negative intervals. Without loss of generality, let us suppose $upper(h) \leq lower(g)$.

By Lemma 4, $d(p,q) \in h$ and $d(p,q) \in g$ are left-definable and right-definable. Since $d(p,q) \in h$ is right-definable, then by Definition 14, there exists an h' such that $upper^+(h) = upper^+(h')$ and $h \subseteq h'$. Since $d(p,q) \in g$ is left-definable, then by Definition 13, there exists an g' such that $lower^-(g) = lower^-(g')$ and $g \subseteq g'$. Then h' and g' are identity or definable intervals. By properties of identity or definable intervals, $lower(g') \leq 4\sigma$, thus, $upper(h') \leq 4\sigma$. By properties of intervals in $DS(\Sigma^+)$, h is right-closed; g is left-open, if $lower(g) \neq 0$. Then all the possible cases where $h \cap g = \emptyset$ are listed below:

- $upper(h) = 0$, $lower(g) \in \{\sigma, 2\sigma, 3\sigma, 4\sigma\}$ or $lower^-(g) = 0$;
- $upper(h) = \sigma$, $lower(g) \in \{\sigma, 2\sigma, 3\sigma, 4\sigma\}$;
- $upper(h) = 2\sigma$, $lower(g) \in \{2\sigma, 3\sigma, 4\sigma\}$;
- $upper(h) = 3\sigma$, $lower(g) \in \{3\sigma, 4\sigma\}$;
- $upper(h) = 4\sigma$, $lower(g) = 4\sigma$.

We can show that given an upper bound or a lower bound of a definable interval, there is a limited number of possibilities of it. For example, if $upper(h') = 2\sigma$, then $h' = [0, 2\sigma]$ or $h' = [0,\sigma] \circ [0,\sigma]$. Thus, there are finitely many possibilities for the corresponding sequences of $d(p,q) \in h$ and $d(p,q) \in g$. By Definitions 13 and 14, every distance constraint in the sequences is in $D(\Sigma^+)$. By Definitions 5 and 8, we can know which LBPT formulas in Σ^+ they come from. For example, if $d(p,q) \in [0, 2\sigma]$ is in $D(\Sigma^+)$ and $p \in points(a)$, $q \in points(b)$, then $NEAR(a,b) \in \Sigma^+$. In each case, we can show $\perp$ is derivable using axioms, which contradicts the assumption that Σ^+ is consistent. Therefore, $D(\Sigma^+)$ is path-consistent. QED.

Theorem 2 *If a finite set of formulas Σ is LBPT-consistent, there exists a metric model satisfying it.*

Proof. Given Σ, by Lemma 1, we can construct an *MCS* Σ^+ containing it. If Σ is LBPT-consistent, so is Σ^+, and hence by Lemma 8 and Lemma 7 there is a metric space (Δ, d) such that all constraints in $D(\Sigma^+)$ are satisfied by d. By Lemma 3, the metric space can be extended to a model M of $B(\Sigma^+)$, thus, of Σ^+ (Definition 5). By properties of maximal consistent sets, for every $\phi \in \Sigma^+$, $\phi \in \Sigma^+ \Leftrightarrow M \models \phi$. Hence, since $\Sigma \subseteq \Sigma^+$, M satisfies all formulas in Σ. QED.

5. DECIDABILITY AND COMPLEXITY OF LBPT

From the bound on the size of the satisfying model, we also have the following theorem:

Theorem 3 *The LBPT satisfiability problem is NP-complete.*

Proof.(sketch) NP-hardness of the LBPT satisfiability problem follows from NP-hardness of the satisfiability problem for propositional logic, which is included in LBPT.

To prove that the LBPT satisfiability problem is in NP, we show that given a finite satisfiable set of LBPT formulas Γ, we can guess a model for Γ and verify that this model satisfies Γ, both in time polynomial in the combined size of formulas occurring in Γ.

The completeness proof shows that, if Γ is consistent, it is satisfiable in a metric model M whose size is polynomially bounded by the number of constants in Γ, and distance function has a fixed finite range. We guess a model like this. To check whether it is a proper model, we need to check whether it is a metric space by Definition 1. This can be done in time which is polynomial in the size of M. To check whether M satisfies Γ, we need to check this for each formula in Γ. This can be done in time which is polynomial in the combined size of formulas in Γ and in the size of M. QED.

6. CONCLUSION

We presented a logic LBPT which formalizes the concepts of being 'possibly part of' (BPT), 'possibly connected' (NEAR) and 'definitely disconnected' (FAR). We provided a sound and complete axiomatistion of it with respect to metric models and showed that its satisfiability problem is NP-complete. An LBPT reasoner is under development and testing, for validating 'sameAs' and 'partOf' matches between spatial objects from authoritative and crowd-sourced geospatial datasets.

References

[1] M. Aiello, I. Pratt-Hartmann, and J. Benthem. *Handbook of Spatial Logics*. Springer, 2007.
[2] E. Clementini and P. D. Felice. Approximate Topological Relations. *International Journal of Approximate Reasoning*, 16(2):173–204, 1997.
[3] H. Du, N. Alechina, M.J. Jackson, and G. Hart. Matching Formal and Informal Geospatial Ontologies. In *Geographic Information Science at the Heart of Europe*, Lecture Notes in Geoinformation and Cartography, pages 155–171. Springer, 2013.
[4] H. Du, N. Alechina, K. Stock, and M.J. Jackson. The Logic of NEAR and FAR. In *Conference On Spatial Information Theory, COSIT 2013*, volume 8116 of *LNCS*, pages 475–494. Springer, 2013.
[5] ISO Technical Committe 211. ISO 19107:2003 Geographic information – Spatial schema. Technical report, International Organization for Standardization (TC 211), 2003.
[6] OpenStreetMap. The Free Wiki World Map. http://www.openstreetmap.org, 2012.
[7] Ordnance Survey. Ordnance Survey. http://www.ordnancesurvey.co.uk/oswebsite, 2012.
[8] Z. Pawlak, L. Polkowski, and A. Skowron. Rough Set Theory. In *Wiley Encyclopedia of Computer Science and Engineering*. 2008.
[9] A.J. Roy and J.G. Stell. Spatial Relations between Indeterminate Regions. *International Journal of Approximate Reasoning*, 27(3):205 – 234, 2001.
[10] S. Schockaert, M. D. Cock, C. Cornelis, and E. E. Kerre. Fuzzy region connection calculus: Representing vague topological information. *International Journal of Approximate Reasoning*, 48(1):314–331, 2008.
[11] T. Williamson. *Vagueness*. Routledge, 1996.
[12] S. Winter. Uncertain topological relations between imprecise regions. *International Journal of Geographical Information Science*, 14(5):411–430, 2000.
[13] F. Wolter and M. Zakharyaschev. Reasoning about Distances. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 1275–1282, 2003.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-101

Computing Optimal Policies for Attack Graphs with Action Failures and Costs

Karel DURKOTA and Viliam LISY
Agent Technology Center, Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague
{karel.durkota, viliam.lisy}@agents.fel.cvut.cz

Abstract. An attack graph represents all known sequences of actions that compromise a system in form of an and-or graph. We assume that each action in the attack graph has a specified cost and probability of success and propose an algorithm for computing an action selection policy minimizing the expected cost of performing an attack. We model the problem as a finite horizon MDP and use forward search with transposition tables and various pruning techniques based on the structure of the attack graph. We experimentally compare the proposed algorithm to a generic MDP solver and a solver transforming the problem to an Unconstrained Influence Diagram showing a substantial runtime improvement.

Keywords. optimal policy, attack graph, markov decision process, and-or graph

Introduction

Attack graphs (AG) are a popular tool for analysing and improving security of computer networks, but they can be used in any domain, where attacks consist of multiple interdependent attack actions. Attack graphs capture all the known sequences of actions that may lead to compromising a system, and they can contain additional information, such as the cost of individual actions and the probability that the actions will be successfully executed. AGs can be used to evaluate risks and design appropriate countermeasures.

In analysis of attack graphs, it is often of interest to identify the optimal strategy of the attacker (i.e., which actions to execute in what situation) and its expected cost. For example, comparing the expected cost of the attack to the expected reward of successfully compromising the target indicates if a rational attacker would attack the system at all [3]. In penetration testing, following the optimal attack strategy can save a lot of valuable time [7]. Computing the optimal strategy for the attacker is also a building block in solving various game-theoretic models of interaction between the attacker and defender of a system. Furthermore, a problem of computing the optimal attack strategy can also be seen as a complex variant of the generic problem of probabilistic and-or tree resolution analysed in AI research [4].

In this paper, we propose an algorithm for computing the optimal attack strategy for an attack graph with action costs and failure probabilities. Unlike previous works assuming that the attack graph is a tree (e.g, [7]) and/or computing only a bound on the actual value (e.g., [3]), we compute the exact optimum and we do not impose any

restriction on the structure of the attack graph. Specifically, our approach allows the attack graph to contain (even oriented) cycles and to have actions with probabilities and costs as inner nodes of the attack graph.

The drawback of our approach is that even a simplified variant of this problem has been shown to be NP-hard in [4]. As a result, we solve it by a highly optimized search algorithm and experimentally evaluate its scalability limitations. We show that the problem can be mapped to solving a finite horizon Markov decision process (MDP) and how the information about the structure of the attack graph can be used do substantially prune the search space in solving the MDP. We compare the proposed approach to recently published method for solving this problem [6] and to a recent version of a generic MDP solver from the International Planning Competition 2011 [5], showing that the proposed method scales orders of magnitude better.

1. Background and Definition

1.1. Attack Graph

AG is a directed graph consisting of two types of nodes: (i) *fact nodes*, that represent facts that can be either true or false, and (ii) *action nodes*, that represent actions that the attacker can perform. Each action has *preconditions* – a set of facts that must be true before action is performed and *effects* – a set of facts that becomes true if action is successfully performed. Moreover, every action has associated probability $p \in (0,1]$ – which denotes the probability that action succeeds and its effects become true, and with probability $1-p$ action fails and attacker cannot repeat this action anymore. We assume that attacker cannot repeat actions for couple of reasons: (i) if actions are correlated and have static dependencies (installed software version, open port, etc.), another attempts to use the same action would result alike, and (ii) if we allow infinitely many repetitions, optimal attack policy (explained further) would collapse into a linear plan with attempting for each action until action succeeds[3]. Finally, each action has associated cost c; if attacker decides to perform action a, he will pay the cost c, regardless whether the action is successful or not.

Definition Let Attack Graph be a 5-tuple $AG = \langle \mathrm{F}, \mathrm{A}, \mathrm{g}, p, \mathrm{c} \rangle$, where:

- F is a finite set of facts
- A is a finite set of actions, where action $a : pre \rightarrow eff$, where $pre \subseteq \mathrm{F}$ is called *preconditions* (we refer to them as $pre(a)$) and $eff \subseteq \mathrm{F}$ is called *effects we refer to them as* $eff(a)$
- $\mathrm{g} \in \mathrm{F}$ is the goal
- $p : \mathrm{A} \rightarrow (0,1]$ is the probability of action to succeed (we use notation p_a for probability of action a to succeed, and with $p_{\bar{a}} = 1 - p_a$ the probability of action to fail)
- $\mathrm{c} : \mathrm{A} \rightarrow \mathbb{R}^+$ is cost of the action (we use notation c_a for cost of the action a).

We use the following terminology: we say that fact f *depends* on action a if $f \in eff(a)$, and similarly, action a depends on f if $f \in pre(a)$

Example of such Attack Graph is in Fig. 1. Diamonds are the inner fact-nodes, that are initially false, but can be activated performing any action on which the facts depend

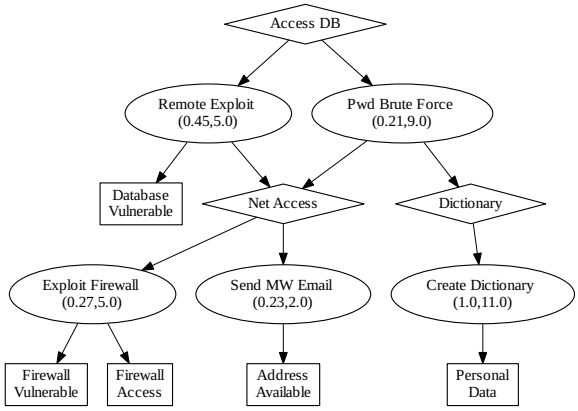


Figure 1. Simple attack graph which shows possible ways how to achieve access to the database (fact node "`Access DB`"). Diamonds are the inner fact-nodes (initially false) that can be turned true, while rectangles are the leaf fact nodes, which are always true. Ellipses depict actions that attacker can perform with success probability p and cost c.

upon. Rectangles represent leaf fact-nodes, that are initially true. Ellipses are the actions that attacker can perform with probability of success p and cost c. In our example we represent action with its name and the couple (p,c). Attacker's goal is to activate fact "`Access DB`" (obtain an access to DB).

The probabilities and costs of the actions can be obtained using Common Vulnerability Scoring System (CVSS)[1] from, i.e., National Vulnerability Database, which scores different properties of vulnerabilities. Probabilities could be computed for example from the access complexities, exploitabilities or availability impacts of the vulnerabilities, whereas costs could be computed from number of required authentication in order to a vulnerability, etc.

1.2. Attack Policy

Solving the AG means to find a policy, that describes what action should attacker perform in every possible evolution of the attack procedure. Fig. 2 depicts optimal policy ξ_{opt} for the problem from Fig. 1, where attacker first attempts to perform action "`Send MW Email`"; if action is successful, he follows the right (sub)policy (solid arc), thus performing action "`Remote Exploit`", otherwise the left (sub)policy (dashed arc), thus action "`Exploit Firewall`", and so on.

Definition An *attack policy* is an oriented binary tree ξ for evaluating attack graph AG. Nodes of ξ are actions, arcs are labeled $+$ or solid line (if parent action was successful) and $-$ or dashed line (if parent action was unsuccessful), and whose leaf-nodes are either $\boxplus$ resp. $\boxminus$ representing successful reps. unsuccessful attack.

[1] www.first.org/cvss

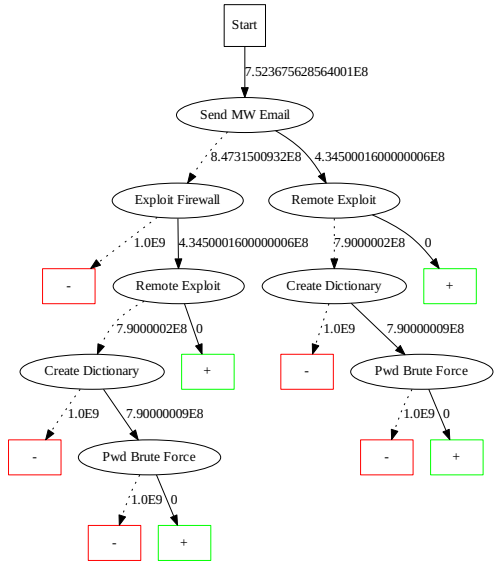


Figure 2. Optimal policy for a simple attack graph from Fig. 1. Attacker should follow solid arcs if previous action was successful, otherwise follow dashed line. Values at arcs represent the expected costs for attacker.

Definition The *expected cost* of an attack policy ξ is a cost over all possible evolutions of the policy. Let ϕ_χ be (sub)tree of ξ rooted at a node χ labeled with an action a, then expected cost of ϕ_χ can be computed recursively as follows:

$$\mathscr{E}(\phi_\chi) = c_a + p_a \times \mathscr{E}(\phi_{\chi^+}) + p_{\bar{a}} \times \mathscr{E}(\phi_{\chi^-})$$

where ϕ_{χ^+} (ϕ_{χ^-}) is the subtree rooted at χ's $+$ branch ($-$ branch) and in the leaf-nodes of the policy is a penalty if attack is unsuccessful $\mathscr{E}(\boxminus) = penalty$ and reward if successful $\mathscr{E}(\boxplus) = reward$.

When we decide which of the two policies, either ϕ_χ rooted at χ labeled with an action a or ϕ_ψ rooted at ψ labeled with action b have lower expected cost, we assume that after performing action a, resp. b, attacker follows an optimal policy. In this case, we override our notation of $\mathscr{E}(\phi_\chi)$ resp. $\mathscr{E}(\phi_\psi)$ to simply $\mathscr{E}(a)$ resp. $\mathscr{E}(b)$.

We assume that our attacker is a *motivated* attacker, that is they continue in attack as long as there are actions that may lead to the goal, regardless of the cost of the attacks. Motivated attacker ceases the attack only when there is no sequence of actions that could result in achieving the goal. Having assumed this type of attacker and the fact that attacks are *monotonic*, meaning that consequence of attack preserves once is achieved [1] (once the fact becomes true, it cannot become false again), it can be shown that every policy, regardless on the order of the action, will have equally the same probability of achieving the goal. Note, that the expected cost of the policy consist of two parts: the probability of achieving the reward or the penalty and the expected cost of the action costs. The probability of successful attack is always the same, thus every policy will have the same

expected cost of the penalty/reward. Thus, it essentially makes no difference whether we choose to reward the attacker for successful attack or penalize for an unsuccessful attack. In fact, distinct policies have different expected costs only because of the different action ordering which imposes different sequences of their costs.

Definition A policy ξ_{opt} is *optimal* if it has the minimal expected cost among all possible policies, thus $\forall \xi \in \Xi : \mathscr{E}(\xi_{opt}) \leq \mathscr{E}(\xi)$, where Ξ is a set of all policies.

Proposition 1.1 *In the optimal policy ξ_{opt} for every (sub)policy ϕ_χ rooted at a node χ labeled with an action a following is true: $\mathscr{E}(\phi_\chi) \leq \mathscr{E}(\phi_{\chi^-})$.*

We will prove it by contradiction. Assume that $\mathscr{E}(\phi_\chi) > \mathscr{E}(\phi_{\chi^-})$ is true. Then due to the monotonicity property the attacker could have followed the (sub)policy ϕ_{χ^-} even before performing action a, which would have saved him the cost of the action c_a. But then this new policy would have had lower expected cost than the policy ϕ_χ, which violates our assumption that ϕ_χ is an optimal policy.

Proposition 1.2 *In the optimal policy ξ_{opt} for every (sub)policy ϕ_χ rooted at a node χ labeled with an action a following is true: $\mathscr{E}(\phi_{\chi^-}) \geq \mathscr{E}(\phi_{\chi^+})$.*

$$\mathscr{E}(\phi_\chi) = c_a + p_a * \mathscr{E}(\phi_{\chi^+}) + p_{\bar{a}} * \mathscr{E}(\phi_{\chi^-}) \tag{1}$$

$$\mathscr{E}(\phi_{\chi^-}) \geq c_a + p_a * \mathscr{E}(\phi_{\chi^+}) + p_{\bar{a}} * \mathscr{E}(\phi_{\chi^-}) \tag{2}$$

$$\mathscr{E}(\phi_{\chi^-}) \geq c_a + p_a * \mathscr{E}(\phi_{\chi^+}) + c_a/p_a \tag{3}$$

1.3. Markov Decision Process

We solve this problem by modeling it as Markov Decision Processes (MDP) [2] which is defined as 4-tuple $\langle S, A, P_\cdot(\cdot,\cdot), R_\cdot(\cdot,\cdot)\rangle$, where:

- S is a finite set states, in our case state is a set of performed actions and label whether the action a was successful (a) or not ($\bar{a}$);
- A is a set of actions, which is equal to the set of actions in the attack graph
- $P_a(s,s')$ is a probability that action a, performed in state s, will lead to state s'; in our case, if action a, with probability p_a is successful, then state $s' = s \cup \{a\}$; if action a is unsuccessful, then state $s' = s \cup \{\bar{a}\}$
- $C_a(s,s')$ is an immediate cost payed after transition to state s' from state s; in our case $C_a(s,s') = c_a$ in all transitions, except when s' is a terminal state, then $C_a(s,s') = c_a - reward$ if goal is achieved in s' and $C_a(s,s') = c_a + penalty$ if goal is not achieved in s'.

Optimal solution is such a policy of MDP, that minimizes an overall expected cost.

2. Algorithm

2.1. Basic Approach

Basic approach is to use MDP with finite horizon, e.g. exhaust every possible action at every decision point and select action having minimal expected cost. In fact,

we use this approach with several pruning techniques which speed up this computation. In Fig. 3 is an example of MDP search of our running example from Fig 1. The root of the MDP is a decision point where we need to decide which of the action among `"Exploit Firewall"`, `"Send MW Email"` and `"Create Dictionary"` is the best to perform. In a naive approach we explore every possible scenario and compute their expected costs $\mathscr{E}($`"Exploit Firewall"`$),\mathscr{E}($`"SendMW Email"`$)$ and $\mathscr{E}($`"Create Dictionary"`$)$. We choose the action with the minimal expected cost.

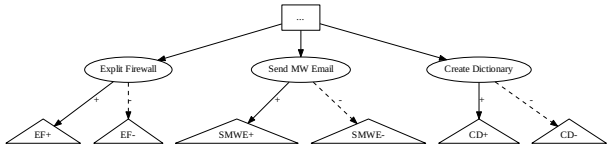


Figure 3. In naive approach we explore every possibility and select action having minimal expected cost.

For performance enhancement, we make use of *transposition tables*, that is: we cache states for which we have computed expected cost and an optimal (sub)policy and reuse these results in future, should we encounter the same state again.

2.2. Sibling-Class Theorem

In [4] authors deal with "probabilistic and-or tree resolution" (PAOTR) problem, mainly for and-or trees with independent tests without preconditions, for which they constructed and proved the Sibling-Class Theorem. Independently, authors in [3] show the same theorem. The Sibling-Class Theorem states, that the leaf-node actions can be grouped into the sibling-classes within which actions' ordering can be determined by simply sorting their R-ratios; hence, no state-search exploration is necessary within sibling-class, only between the sibling classes. Two actions belong to the same sibling class if they have common parent in the and-or tree. As they consider inner nodes to be either AND or OR node, naturally there are two types of sibling classes: the AND-sibling classes and the OR-sibling classes. R-ratios of an action is computed as follows:

$$R(a) = \frac{p_a}{c_a} \text{ if action } a \text{ is in OR-sibling class} \tag{4}$$

$$R(a) = \frac{p_{\bar{a}}}{c_a} \text{ if action } a \text{ is in AND-sibling class} \tag{5}$$

Conjecture 2.1 *Sibling-Class Theorem for and-or trees without preconditions can be applied to an and-or graph with precondition using following rules for creating OR-Sibling Classes:*

- *actions a and b belong to the same OR-Sibling class* iff*:* $pre(a) = pre(b) \wedge |pre(a)| = |pre(b)| = 1$.

and following rules for AND-Sibling Class:

- *action a and b belong to the same AND-Sibling class* iff*:* $pre(a) \neq pre(b) \wedge |pre(a)| = |pre(b)| = 1 \wedge (\exists c \in A : pre(a) \in \mathit{eff}(c) \wedge pre(b) \in \mathit{eff}(c))$.

Action $a \in A$ *cannot be pruned* iff: $|pre(a)| > 1 \vee (\exists c_1, c_2 \in A : c_1 \neq c_2 \wedge pre(a) \in \mathit{eff}(c_1) \wedge pre(a) \in \mathit{eff}(c_2))$.

The Sibling Theorem is proved only for and/or trees, while we empirically checked and use it for and/or graphs.

Example Assume we have the same problem as in Fig. 3 and we come to the same decision point as previously. But now we computed $R(\texttt{"Exploit Firewall"}) = \frac{0.27}{5.0} = 0.054$, $R(\texttt{"Send MW Email"}) = \frac{0.23}{2.0} = 0.115$ and $R(\texttt{"Create Dictionary"}) = \frac{1.0}{11.0} = 0.091$ and we know that actions `"Exploit Firewall"` and `"Send MW Email"` have the same parent node `"Net Access"`, thus belong to the same OR-sibling class, while action `"Create Dictionary"` belongs to a separate sibling class. Now we explore only actions that have maximum R-ratios in each sibling class, thus only actions `"Send MW Email"` and `"Create Dictionary"`, and action `"Exploit Firewall"` surely will not be the first action in the optimal policy. Fig. 4 depicts nodes that must be explored (white nodes), and nodes that are pruned (grey nodes).

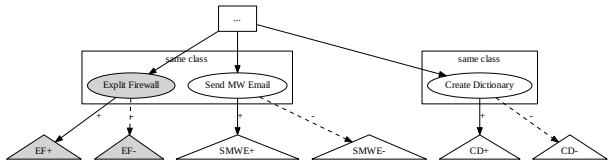


Figure 4. This figure presents nodes that are explored (white) and nodes that can be pruned (grey) in the MDP search if we know that actions `"Exploit Firewall"` and `"Send MW Email"` are in the same sibling class and action R-ratios.

2.3. Branch and Bounds

As another pruning technique we use branch and bounds. For this technique we reuse previously computed expected costs of the subtrees of the MDP to prune future subtrees if know that an optimal solution cannot exist there. Specifically, when we face the decision either utilize (sub)policy ϕ_a – starting with an action a – or (sub)policy ϕ_b – starting with action b – we choose policy ϕ_b only if $\mathscr{E}(\phi_b) < \mathscr{E}(\phi_a)$, implying:

$$\mathscr{E}(\phi_b) < \mathscr{E}(\phi_a) \tag{6}$$

$$c_b + p_b * \mathscr{E}(\phi_{b^+}) + p_{\bar{b}} * \mathscr{E}(\phi_{b^-}) < \mathscr{E}(\phi_a) \tag{7}$$

$$c_b + p_b * \mathscr{E}(\phi_{b^+}) + p_{\bar{b}} * \mathscr{E}(\phi_{b^+}) < \mathscr{E}(\phi_a) \tag{8}$$

$$c_b + \mathscr{E}(\phi_{b^+}) < \mathscr{E}(\phi_a) \tag{9}$$

$$\mathscr{E}(\phi_{b^+}) < \mathscr{E}(\phi_a) - c_b \tag{10}$$

where from (8) to (9) we used property of the optimal policy that $\mathscr{E}(\phi_{b^+}) \leq \mathscr{E}(\phi_{b^-})$, and then fact that $p_b + p_{\bar{b}} = 1$. Thus, $\mathscr{E}(\phi_a) - c_b$ is an upper-bounds for $\mathscr{E}(\phi_{b^+})$. If anytime

during the computation it exceeds this bound, we can immediately stop the computation of the b^+ branch.

Similarly, having computed the $\mathscr{E}(\phi_b)$, we can bound again the branch $\mathscr{E}(\phi_{b^-})$ as follows

$$\mathscr{E}(\phi_b) < \mathscr{E}(\phi_a) \tag{11}$$

$$c_b + p_b * \mathscr{E}(\phi_{b^+}) + p_{\bar{b}} * \mathscr{E}(\phi_{b^-}) < \mathscr{E}(\phi_a) \tag{12}$$

$$p_{\bar{b}} * \mathscr{E}(\phi_{b^-}) < \mathscr{E}(\phi_a) - c_b - p_b * \mathscr{E}(\phi_b) \tag{13}$$

$$\mathscr{E}(\phi_{b^-}) < (\mathscr{E}(\phi_a) - c_b - p_b * \mathscr{E}(\phi_b))/p_{\bar{b}} \tag{14}$$

Example Assume different example, where we face the problem of choosing the best (sub)policy ϕ_a, ϕ_b, ϕ_c and ϕ_d. Branches $\mathscr{E}(\phi_{a^+})$ and $\mathscr{E}(\phi_{a^-})$ we must compute to obtain $\mathscr{E}(\phi_a)$. Next, we compute expected cost $\mathscr{E}(\phi_{b^+})$ and assume that it turns out to be higher than $\mathscr{E}(\phi_a)$, hence, we can prune the computation of the branch b^-, as action b will never have less expected cost then action a. Next, let's say $\mathscr{E}(\phi_{c^+})$ in the branch c^+ turned out to be lower than $\mathscr{E}(\phi_a)$. It means that we can upper bound the $\mathscr{E}(\phi_{c^-})$ by $\frac{\mathscr{E}(\phi_a) - c_c - p_c * \mathscr{E}(\phi_c)}{p_{\bar{c}}}$. Let's say that $\mathscr{E}(\phi_{c^-})$ obeyed the bound, thus, action c is the best action that attacker can perform so far. Note, that it is unnecessary to compute total expected cost of $\mathscr{E}(\phi_c)$ to determine that it is lower then $\mathscr{E}(\phi_a)$, it is direct implication from the fact that it satisfied both bound conditions. Finally, during the computation of branch d^+ it turned out to violate new upper bound $\mathscr{E}(\phi_c) - c_d$, thus its computation was terminated and branch d^- was pruned as well.

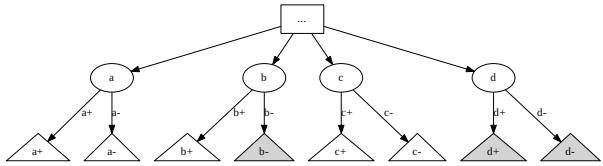


Figure 5. This figure presents nodes that are explored (white) and nodes that can be pruned (grey) due to the branch and bound technique.

2.4. Heuristics

Finally, we designed heuristic approach to compute the lower bound of the expected cost of an attack graph by setting costs of the actions to zero and taking into account only the actions' probabilities. This relaxation gives us freedom in action ordering as any (valid) ordering will produce exactly the same probability of success of the policy and thus the overall expected cost. We use this heuristics in two ways: (i) if computed heuristics exceeds the given upper bound, this branch of computation can be pruned, and (ii) according the heuristics we order the action in which we compute remaining actions'

expected costs. If we start with the most promising action, more of the future branches might get pruned.

Nevertheless, we came across an issue in this approach due to the fact that our attack representation is not a tree but a directed cyclic graph. Which means that performing an action can be beneficial in several possible branches at once if action has more then one root-node paths in the attack graph, which results, that the probability of the action will be count multiple times into the overall probability of success is increased. This causes expected cost, computed as $(1 - probability) * penalty$ to decrease. Since we minimize the expected cost this issue keeps the heuristics still admissible.

3. Experiments

We experimentally compared our algorithm with two other approaches, namely Unconstrained Influence Diagrams, and using probabilistic planner from International Planning Competition.

3.1. Guido approach

This approach, described in [6], converts an attack graph into an Unconstrained Influence Diagrams (UID) — a graphical representation of a decision situations using probabilistic interference — upon which existing solvers can be run. We ran a Guido solver as described in the article. This approach showed to be insufficiently scalable for the problems with large (>20 actions) attack graphs.

3.2. Probabilistic planning

As an other approach, we decided to use a domain independent probabilistic planner SPUDD that competed in International Plannign Competition (IPC) in 2011. SPUDD is based on iterative value computation of MDP and uses own specification language. Since it computes MDP, it needs to have set either discount factor $\gamma = [0,1)$, or $\gamma = 1$ and the horizon set to an integer. For our purposes, discount factor γ must be set to 1, hence horizon had to be chosen appropriately. To ensure that SPUDD finds an optimal solution, we chose to set the horizon to number of actions in the attack graph.

3.3. Experiment settings

We experimentally ran and compared our algorithm DynProg, Guido and SPUDD approaches on the three different realNetwork frameworks with different configurations. We ran experiments on Intel 3.5GHz with memory resource up to 10GB. In DynProg we set the $penalty = 10^9$ and $reward = 0$. In Tab. 1 we present running times of each approach.

4. Conclusion and Future Works

Our algorithm showed to outperform other two approaches in time complexity and scalability. Unfortunately, it ofter runs out of the memory due to the transposition tables and

Problem	DynProg [ms]	Guido [ms]	SPUDD [ms]
Local+2	**51**	85	1000
Local+3	**155**	546	11000
Local+4	**443**	76327	70000
Local+5	**5389**	(OoM)	656000
Local+6	(OoM)	(OoM)	**6152000**
Cross2	**4**	408	1000
Cross3	**38**	23796	9000
Cross4	**504**	(OoM)	287000
Cross5	**3587**	(OoM)	8373000
Cross6	**60351**	(OoM)	(OoT)
LocalChain3-3	**0**	9	**0**
LocalChain4-4	**0**	70	1000
LocalChain5-5	**0**	1169	3000
LocalChain6-6	**0**	17133	23000

Table 1. Time comparison of DynProg, Guido and SPUDD approaches over three types of problems with different complexities. Shortcuts: (OoM) - Out of Memory, (OoT) - Out of Time ($> 10^7$ms).

very large search state-space anyway. Other optimizations can be proposed, as better representation of a state or more accurate heuristics for better pruning. This algorithm can be used in game theoretic manner in couple of ways. Here we present two directions: (i) determine what honeypot configurations maximize the probability that an attacker would be detected during their attacks on the realNetwork and (ii) for security hardening determining which subset of vulnerabilities should administrator fix in order to secure the realNetwork, that is, that for the attacker it is not worth to attack to begin with.

Acknowledgement

This research was supported by the Office of Naval Research Global (grant no. N62909-13-1-N256) and Czech Ministry of Interior grant number VG20122014079.

References

[1] Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based realnetwork vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 217–224. ACM, 2002.

[2] Richard Bellman. Dynamic programming and lagrange multipliers. *Proceedings of the National Academy of Sciences of the United States of America*, 42(10):767, 1956.

[3] Ahto Buldas and Roman Stepanenko. Upper bounds for adversaries utility in attack trees. In Jens Grossklags and Jean Walrand, editors, *Decision and Game Theory for Security*, volume 7638 of *Lecture Notes in Computer Science*, pages 98–117. Springer Berlin Heidelberg, 2012.

[4] Russell Greiner, Ryan Hayward, Magdalena Jankowska, and Michael Molloy. Finding optimal satisficing strategies for and-or trees. *Artificial Intelligence*, 170(1):19–58, 2006.

[5] Jesse Hoey, Robert St-Aubin, Alan J Hu, and Craig Boutilier. Spudd: Stochastic planning using decision diagrams, 1999.

[6] Viliam Lisý and Radek Píbil. Computing optimal attack strategies using unconstrained influence diagrams. In *Intelligence and Security Informatics*, pages 38–46. Springer, 2013.

[7] Carlos Sarraute, Gerardo Richarte, and Jorge Lucángeli Obes. An algorithm to find optimal attack paths in nondeterministic scenarios. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, AISec '11, pages 71–80, New York, NY, USA, 2011. ACM.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-111

Semantifying Triples from Open Information Extraction Systems

Arnab DUTTA [1], Christian MEILICKE and Heiner STUCKENSCHMIDT
Data and Web Science Research Group, University of Mannheim, Germany

Abstract. The last few years have witnessed some remarkable success of the state-of-the art unsupervised knowledge extraction systems like NELL and REVERB. These systems are gifted with typically web-scale coverage but are often plagued with ambiguity due to lack of proper schema or unique identifiers for the instances. This classifies them apart from extraction systems like DBPEDIA, YAGO or FREEBASE which have precise information content but have smaller coverage. In this work we bring together the former to enrich the later with high precision novel facts and present a statistical approach to discover new knowledge. In particular, we semantify NELL triples using DBPEDIA.

Keywords. open information extraction, information integration, knowledge generation, statistical modeling.

1. Introduction

With the growing popularity of unsupervised techniques of knowledge extraction from text, web corpora, there is an advent of a new genre of extraction systems commonly know as *open* information extraction (OIE) systems. ”Open” in the sense, they are not limited to Wikipedia or any specific resource but the whole of web. Systems like NELL [5], REVERB [11] have gained a quick prominence marked by their web scale coverage and huge fact base. However, such systems often lack a schema and hence, is difficult to correctly disambiguate entities from the OIE fact base. On the other hand, knowledge bases (KBs) like DBPEDIA [1], YAGO [19], FREEBASE [2] mark another class of extraction systems which are of higher quality, precise and maintain a well-structured schema but at the expense of a poor coverage (often restricted only to Wikipedia).

There is considerable potential in exploiting the data maintained by OIE for analyzing, reasoning about, and discovering novel facts and generation of web search engines [9]. In this work we integrate these two broad domains in a symbiotic fashion; OIE systems exploit the clean ontological structure of closed IE systems, thereby imparting uniqueness to its entities; and the closed IE systems enrich themselves from the broader coverage of the OIE. We present a methodology to automatically find new DBPEDIA triples by exploiting the information content in NELL. To achieve this, we need a synergistic integration of solving the three major sub tasks:

[1]Corresponding Author: Arnab Dutta, University of Mannheim, 68159 Mannheim, Germany; E-mail: arnab@informatik.uni-mannheim.de

1. how to precisely find the references of the instances within an OIE triple, to a closed KB instance (DBPEDIA in our case). This resembles with the task of entity linking where a term in a text is linked to a KB entry. But the lack of any context for OIE triples sets the scenario different for us and deters us to use any of-the-shelf entity linking tool.
2. assuming we have correctly deciphered the references, how can we map the relationship within an OIE triple to an analogous closed domain property.
3. how can the instance and property matching interplay to enrich a closed KB.

Let us consider a NELL triple of the form *stadiumlocatedincity*(*riverfest*, *little_rock*), where two entities are in a semantic relationship defined by *stadiumlocatedincity*. Even though we might have an intuitive understanding of the property, it is difficult to interpret the exact real world entities the terms are referring to; *little_rock* can refer to a range of cities in USA or a person or even a US naval ship. The problem gets more complicated since, unlike the well defined properties in DBPEDIA or YAGO, the OIE properties often lack strict domain and range definitions. This makes it difficult to determine what fits in as a subject and object. Note, NELL has its own schema but in an attempt to propose a general solution, we keep our approach agnostic to this information.

We propose a *learn-and-fill* approach to use the ambiguous triples in order to generate new facts. First, we map the NELL instances to DBPEDIA instances (Section 2.1) using a probabilistic approach. This is not the core contribution of this paper and has been already addressed before [7]. Second, we use the mappings to look for a semantic relationship (clean infobox properties with "/ontology" namespace) in DBPEDIA (Section 2.2) and use it as a likely predictor for the NELL property. For instance, we *learn* from other NELL triples with same property, like *stadiumlocatedincity*(*meyerson_symphony_center*, *dallas*) and many more that `location` can be a likely mapping. The final piece of the solution lies in integrating these two mapping solutions to generate new facts (Section 2.3). We can now *fill in* the original NELL triple with the learnt property `location` to generate a new facts like `location(Cincinnati_Bell/WEBN_Riverfest, Little_Rock,_Arkansas)`. We apply statistical techniques for the purpose and show its impact in generating high quality facts (Section 3).

2. Methodology

2.1. Mapping Instances

We map the individual subject and object occurring within a NELL triple to DBPEDIA instances. In this regard, we explore two different methods. The first and a naive approach is to look for the most frequent sense of the terms occurring in NELL triples by exploring intra-Wikipedia page links connecting anchor texts to their respective pages. A detailed analysis of this approach can be found in [8]. It is a simple approach, without exploiting any contextual information to improve the mappings.

As an improvement, we incorporate the type information of the mapped DBPEDIA instances. We initiate with the most frequent instance mappings from NELL to DBPEDIA and let them to determine the possible types of DBPEDIA instances *allowable* in the context of the given NELL property. Subsequently, the type information guides the mapping selection process again. These bootstrapped approach of selecting-and-refining is solved

using Markov Logic Networks [17] and leads to better results [7] than the naive way. We employ this technique to generate a refined set of hypotheses where every NELL instance eventually has atmost one mapping to a DBPEDIA instance.

2.2. Mapping Properties

This section presents our approach for mapping an OIE property to an analogous DBPEDIA property. For every NELL triple of the form $n_p(n_s, n_o)$ we map the subject (n_s) and object (n_o) individually to DBPEDIA instances d_s and d_o respectively (refined instance mappings from the probabilistic framework [7]). Using a DBPEDIA SPARQL endpoint, we query[2] for a possible property d_p involved in some triple of the form $d_p(d_s, d_o)$. If such a triple exists, then d_p can be considered as a likely DBPEDIA mapping of n_p. We apply this technique over all the NELL properties. Furthermore, we also consider inverse property mappings. We denote d_p^{inv} to be an inverse mapping for n_p, if the triple d_p^{inv} (d_o, d_s) exists in DBPEDIA. The methodology proposed in this work is applicable for both the two cases and we use d_p as a general notation for DBPEDIA property.

Likelihood Estimate: We want to estimate the likelihood of every possible mapping of n_p to one or more d_p. A naive frequency count of the mappings can give us a likelihood estimate. For instance, if NELL property *bookwriter* is mapped to `author` in k out of n cases and to `writer` in (n-k) out of n cases, then the likelihood of the mapping *bookwriter* to `author` is $\frac{k}{n}$, and to `writer` is $\frac{(n-k)}{n}$. Finally, selecting candidates above a threshold score, could be a simple solution. However, this approach suffers from two major drawbacks: first, any conceptually similar property (as in this case) might be eliminated out due to lack of sufficient evidence (low likelihood score); second, finding a correct threshold. An improved approach could be to incorporate the type information of the mapped DBPEDIA instances as well.

Formally, we define a set T_{n_p} consisting of NELL triples with property n_p. For each such triple, we collect the type of the mapped DBPEDIA subject and object, denoted by $dom(d_s)$ and $ran(d_o)$ respectively. Now, querying for triples like $d_p(d_s, d_o)$ can have the following possibilities:

1. returns an empty set, indicating absence of any d_p. This can happen if there is no such triple in DBPEDIA or the mapped instances are wrong at the first place.
2. returns a single possible value for d_p (e.g. *airportincity*(*helsinki_vantaa_airport*, *helsinki*) maps to `city`(`Helsinki_Airport`, `Helsinki`))
3. returns multiple values for d_p. (e.g. *airportincity*(*vnukovo*, *moscow*) maps to `city` (`Vnukovo_International_Airport`, `Moscow`) and `location`(`Vnukovo_International_Airport`, `Moscow`))

Case (2) and Case (3) are given an unified representation by framing discrete associations as, $\{n_p, d_p, dom(d_s)\}$ and $ran(d_s)$ (refer to Table 1). Hence, the example in Case (3) translates to {*airportincity*, `city`, `Airport`, `Place`} and {*airportincity*, `location`, `Airport`, `Place`}. Case (1) is not translated. All the associations for n_p thus formed is denoted as A_{n_p}. It is important to note that, $|T_{n_p}| \leq |A_{n_p}|; \forall n_p \in T_{n_p}$. A blank value ('-') is attributed to a missing instance type in DBPEDIA or non-mappabiliy of n_p due to reasons mentioned in Case (1) above.

[2] select ?d_p where {d_s ?d_p d_o}

Table 1. A snippet of the actual associations presenting a positive example with *airportincity* and a negative example with *agentcreated*. Missing value is marked with "-".

A_{n_p}	n_p	K_{n_p}	i	d_p^i	$dom(d_s^i)$	$ran(d_o^i)$	$\tau_{n_p}^i$	$\hat{\tau}$
$\wedge \ldots A_{airportincity} \ldots \vee$	*airportincity*	55%	1	location	MilitaryStructure	-	150	1.21
			2	location	Airport	-	**0.86**	1.21
			3	isPartOf	Settlement	Settlement	150	1.21
			4	isPartOf	Settlement	-	37.5	1.21
			5	city	Airport	-	**0.86**	1.21
			...	...	...	...	...	...
$\wedge \ldots A_{agentcreated} \ldots \vee$	*agentcreated*	9%	1	notableWork	Writer	Play	496.6	3.12
			2	notableWork	Writer	TelevisionShow	3973	3.12
			3	occupation	Person	Book	12.7	3.12
			4	occupation	Settlement	-	37.5	3.12
			5	knownFor	Scientist	Book	3973	3.12
			...	...	...	...	...	...

Now, we introduce K_{n_p}, the mapping factor determining the degree to which a particular NELL property can be mapped, as

$$K_{n_p} = \frac{\sum_{j=1}^{|T_{n_p}|} C(j)}{|T_{n_p}|}; \quad where\ C(j) = \begin{cases} 1; \textit{atleast one property mapping for } n_p \textit{ in } T_{n_p}^j \\ 0; \textit{otherwise} \end{cases}$$

Assuming, we have only ten triples for *airportincity*, eight have been mapped (mixture of Case (2) and (3) above), two have been not (Case (1) above), then $K_{airportincity} = \frac{8}{10}$. Here, under the column K_{n_p} we present the actual value which is 0.55. Then we apply an association rule [14] of the form $\{n_p \Rightarrow dom(d_s^i), ran(d_o^i)\}$, on A_{n_p}. This means, if the NELL property is n_p then the type of the mapped DBPEDIA subject instance is $dom(d_s^i)$ and type of the object instance is $ran(d_o^i)$. We compute the confidence, denoted as $conf$, for each such rule, and which denotes the frequency of co-occurrence of $dom(d_s^i)$ and $ran(d_o^i)$, whenever n_p occurred. Hence, the confidence for the i^{th} association for a property is denoted as $conf_{n_p}^i$, and defined as,

$$conf(n_p \Rightarrow (dom(d_s^i), ran(d_o^i))) = conf_{n_p}^i = count(n_p, dom(d_s^i), ran(d_o^i)) \Big/ |A_{n_p}|$$

Referring to Table 1, $conf_{agentcreated}^3 = count(agentcreated, Person, Book)/|A_{agentcreated}|$. Note the *count* function is not just the frequency count of the joint occurrence of a particular n_p and its associated DBPEDIA domain and range values, but, also the sub-classes of each of the domain and range. The rational is, if we observe an association like *agentcreated* $\Rightarrow$ (`Person`, `Book`) then any other association like *agentcreated* $\Rightarrow$ (`Scientist`, `Book`) should also be considered as an evidence for the former association. `Scientist` being a sub-class of `Person` in the DBPEDIA ontology, is not a different association but a more particular case. Finally, each association, is awarded with a confidence of $conf_{n_p}^i$.

We combine K_{n_p} and $conf_{n_p}^i$ to define the second factor called τ (*tau*) defined as,

$$\tau^i_{n_p} = \frac{(1 - K_{n_p})}{conf^i_{n_p}};\ \forall i \in A_{n_p}$$

This quantifies the badness of a particular association for a particular n_p with mapping factor K_{n_p}. A low confident association with low K_{n_p} will give a high $\tau^i_{n_p}$ ($\tau^3_{airportincity}$ in Table 1) while, a more confident association with high K_{n_p} minimizes the ratio, hence less bad ($\tau^5_{airportincity}$ in Table 1). We are primarily interested in the later ones. We employ, $\tau^i_{n_p}$ as our unit of measurement and define a minimum threshold, $\tau^{min}_{n_p}$ which occurs when $conf^i_{n_p}$ attains the maximum confidence (follows directly from the definition of tau above). Intuitively, $\tau^{min}_{n_p}$ defines the upper bound for the best score possible.

Threshold Learning: In this section we devise a technique to learn a correct threshold for $\tau^i_{n_p}$. Our objective is to solve the problem with least number of parameters possible. There can be three broad association patterns possible:

- a single high $conf^i_{n_p}$ association, among many others
- multiple closely spaced possible DBPEDIA properties with almost same $conf^i_{n_p}$
- No clear winner, but multiple candidates with low $conf^i_{n_p}$

We aim at modeling these different scenarios which would select the first two cases but not the third one. The rational is, any association rule with a low confidence is not an apt predictor for n_p. In this regard, we observed that the underlying data set had a distribution pattern over K_{n_p} (detailed figures in Section 3). We use it to manually determine a threshold for K_{n_p}, denoted as K_{thres}. Hence, we select data points having K_{n_p} atleast K_{thres}. This gives us a set of co-ordinates, D given as $\{\ldots, (K_{n_p}, \tau^{min}_{n_p}), \ldots\}$. We fit a linear regression model on set D, motivated by the fact that τ shows a linear dependence on K_{n_p} (our initial analysis had revealed that introducing $conf$ as another variable had minimal effect on τ, hence we use it as a constant). With such a linear predictive analysis method, we can have an estimate of τ, defined as $\hat{\tau}$ for every K_{n_p}. Note that, we trained our model using the data points attained using the maximum confidence (analogously $\tau^{min}_{n_p}$), hence, the linear model is an automatically learnt threshold on τ_{n_p}. We use $\hat{\tau}$ to compare with every $\tau^i_{n_p}, \forall\, i \in A_{n_p}$. Some scores fall below the prediction (the likely associations) and some are way above it (less likely ones) (refer to Table 1, correct association values are marked in bold). The likely associations allow us to select the final DBPEDIA property driven by the rule $(dom(d^i_s), ran(d^i_o)) \Rightarrow d^i_p$. Note that, in determining the property match we exploited the class information and remained un-informed of the actual DBPEDIA property involved. Analyzing the patterns now,

- Multiple associations but a single one with a high $conf^i_{n_p}$. This makes $\tau^i_{n_p} \simeq \hat{\tau}$
- Multiple closely placed associations with almost same $conf^i_{n_p}$, making $\tau^i_{n_p} \simeq \tau^j_{n_p} \simeq \hat{\tau}$; $i \neq j$. (refer Table 1, $\tau^2_{airportincity}$ and $\tau^5_{airportincity}$)
- No clear winner, but multiple candidates with low $conf^i_{n_p}$ making $\tau^i_{n_p} \ggg \hat{\tau}$ (refer Table 1, $\tau^1_{agentcreated}$, $\tau^5_{agentcreated}$)

2.3. Knowledge Generation

As a final module, we combine our two solutions in an attempt to generate new facts missing in DBPEDIA. Reiterating, we have a set of NELL triples, for which we could

Table 2. Precision of knowledge generated, for the properties predicted by our approach. * denotes inverse property mappings. Best results marked in bold

n_p	d_p	p_{inst}	p_{prop}	p_{fact}
headquarteredin	`headquarter`	0.96	1.0	**0.93**
visualartistartform	`movement`	0.99	0.06	0.057
	`field`	0.99	0.95	**0.93**
personhasresidenceincountry	`nationality`	1.0	0.33	**0.33**
airportincity	`city`	0.63	1.0	**0.3**
	`location`	0.50	1.0	0.17
stadiumlocatedincity	`location`	0.95	1.0	**0.91**
televisionstationincity	`locationCity`	0.98	1.0	**0.96**
	`location`	0.38	1.0	0.0
televisionstationaffiliatedwith	`broadcastNetwork`	0.99	1.0	0.98
	`formerBroadcastNetwork`	0.995	1.0	**0.99**
radiostationincity	`broadcastArea`	1.0	1.0	**0.90**
	`city`	0.50	1.0	0.0
personhasethnicity	`deathPlace`	0.70	0.0	0.0
	`birthPlace`	0.70	0.60	**0.20**
haswife	`partner`	0.96	1.0	**0.92**
	`spouse`	0.96	1.0	**0.92**
*musicianinmusicartist**	`bandMember`	0.98	1.0	**0.96**
	`associatedMusicalArtist`	0.50	1.0	0.0
*agentcreated**	`author`	1.0	0.80	**0.80**
*citycapitalofcountry**	`largestCity`	1.0	0.91	0.91
	`capital`	1.0	1.0	**1.0**
*automakerproducesmodel**	`manufacturer`	0.75	1.0	**0.50**
Macro-average	-	0.97	0.96	0.77

map its instances to DBPEDIA and its properties to analogous DBPEDIA properties. This provides strong evidence for the portion of NELL triples for which the property could not be mapped. This fraction is given by $1 - K_{n_p}$. Having n_p successfully mapped to d_p, we can use d_p to *fill-in* the missing relationships between the DBPEDIA instances for these non-mapped triples. Hence, the fraction of non-mapped triples for a NELL property defines an upper bound on the scope for new facts generation. This is a strict upper bound, since, a NELL triple can be non-mapped due to either:

1. a missing semantic relation between two correctly mapped DBPEDIA instances
2. a missing semantic relation between incorrectly mapped DBPEDIA instances
3. there is no mapping of instance possible at the first place

Clearly, points (1) and (2) are the ones which can lead to knowledge generation. As a matter of fact, Case (2) will generate wrong facts (further details in Section 3.1) and Case (3) cannot be dealt with our approach. In this respect, it is interesting to note a dilemma: if a property is nearly 100% mappable, we are more confident with its evidences but it gives us lesser scope of knowledge generation and vice versa.

3. Empirical Results

We use the NELL data set, having approximately 1.9Mi triples, for our experiments, but without the triples with property *generalizations* since it is analogous to `rdf:type` which expresses class instantiation. In this paper, we are focusing on the more complex task of finding the correct mapping for a potentially ambiguous property from NELL instead of generating facts like *isA(london, city)*. Note that mapping the instances, simultaneously solves the problem of determining the class of those instances since most of them are already typed in DBPEDIA.

3.1. Performance

Next we compute the precision of the newly generated triples. Since, a gold standard is not available, we resort to manual annotation scheme. Three annotators were provided samples of 300 NELL triples each. Annotators marked every mapping of the subject, property and object as "Correct" or "Incorrect" and also marked the original NELL triple to be "Correct", "Incorrect" or "Ambiguous". The later annotation was important since, even if the mapping of instances and properties are accurate, a wrong NELL triple in the first place will still lead to a wrong fact generated. Based on this agreement, only the triples with correct instance and property matches were considered as *true positives*. Even if one of the instances or the property match was incorrect, the triple was marked as a *false positive*. In Table 2 we present the precision scores for instance mappings (p_{inst}), property mappings (p_{prop}) and generated facts (p_{fact}). Note that, inaccurate instance mappings often lead to lower fact precision even if property mapping was precise (exactly the Case (2) mentioned in Section 2.3). This happens with {*airportincity*, `location`}. Also, the other way round, lower p_{prop} minimises p_{fact} inspite of a high p_{inst} as seen with {*visualartistform*, `movement`}. Hence, a highly accurate p_{inst} and p_{prop} together contributes to a high p_{fact}. The NELL properties with an asterisk (*) denote the inverse properties learnt and likewise present the precision of new triples. Observe that for some properties, we have dual choices of d_p, *visualartistartform* for instance. When mapped to `field` the precision of new triples were better than when mapped to `movement`, even though the later fitted the domain/range restrictions but when used in fact generation, led to senseless triples. Hence annotators marked it as a false positive. The line of reasoning is similar for *personhasethnicity* which had `birthplace` as the mapped property. On the other hand, `largestCity` was accepted as a mapped property for *citycapitalofcountry* since the triples generated as a whole were correct. Overall, we had a precision of 0.97 for instance mappings, 0.96 for property mappings, giving us 0.77 as macro-averaged precision for the generated facts.

3.2. Regression Analysis

In Figure 1(a, c) we present the distribution of $\tau_{n_p}^{min}$ over K_{n_p} (defined in Section 2.2), both for the direct and inverse property mappings respectively. We observe a similar trend in both the figures in the sense that higher values are attained for poorly mapped properties and properties with higher K_{n_p} tend to have lower values for $\tau_{n_p}^{min}$. This allows us to select the points on and beyond a particular threshold (in our case, $K_{thres} = 35\%$). In Figure 1(b,

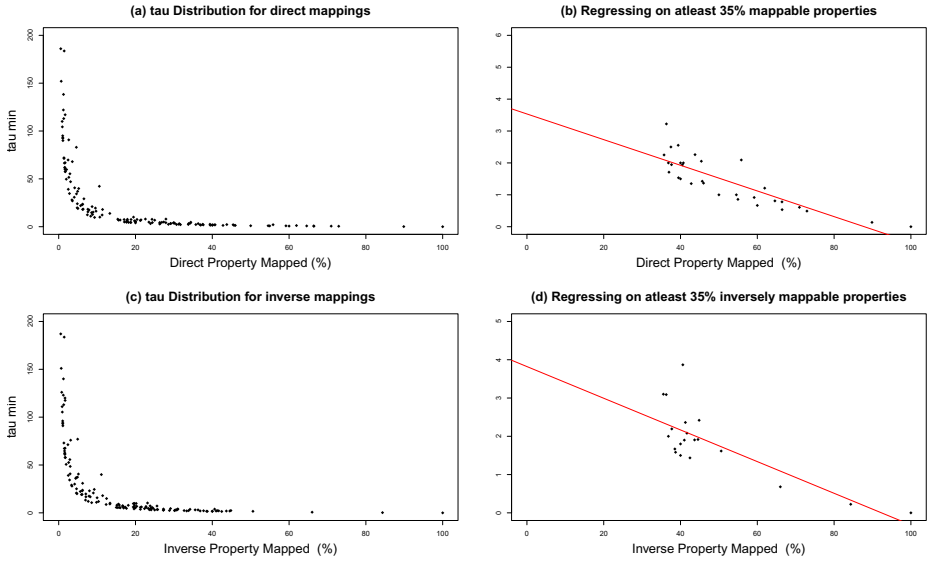


Figure 1. Regression analysis of direct and inverse property mappings.

d), we *zoom in* the data points beyond 35%, where we observe a linear variation. These points comprise of the set D (defined in 2.2). Likewise, we use these as training points to fit a linear predictive model having a single independent variable K_{n_p}. This is shown with the line fitting the points such that the squared error loss is minimized. The linear dependence relation between K_{n_p} and τ allows for this design choice. The regression line sets a self adjusting threshold varying across properties.

Furthermore, in Table 3 we present few examples which shows an interesting aspect of our method. The column labeled n_p denotes the NELL property and d_p the analogous DBPEDIA property learnt. The data is interpreted as follows: when the NELL property *headquarteredin* was mapped to `headquarter`, 39.4% of the mapped subjects were of type `Airline`, 16.6% were `Monarch` and analogously for the range values. The interesting aspect of the approach is that we are able to conserve the fine grained information latent in the facts and not just broadly classify them with some top level concepts.

4. Related Work

Matching Candidates: Seminal work include contributions by Bunescu and Paşca [4] and Cucerzan [6] who focused on the usage of Wikipedia categories and global contexts, respectively. The Silk framework [20] discovers missing links between entities across linked data sources by employing similarity metrics between pairs of instances. In contrast to these approaches, our method employs the most frequent sense of a term from Wikipedia. We combine this information together with the type-information from DBPEDIA in order to automatically refine the entity references [7].

Matching Properties: Much work has been done in the area of aligning ontologies of which PARIS [18] requires special mention which performs probabilistic alignment of relations, instances and schema across ontologies.

Table 3. Domain and Range distribution of mapped NELL properties for the set of new facts generated.

Domain (%)	n_p	d_p	Range (%)
Company(33.33)	*newspaperincity*	headquarter	City(50) Administrative-Region(33.33)
Airline(39.4) Company(36.6) BroadcastNetwork(5.6)	*headquarteredin*	headquarter	City(46.1) Settlement(37.6) Administrative-Region(11.3)
OfficeHolder(50) Person(33.3) Monarch(16.6)	*personhasethnicity*	birthPlace	Country(100)
OfficeHolder(40) SoccerManager(30) Model(10)	*personhas-residenceincountry*	nationality	Country(100)
Television-Station(73.8) RadioStation(26.2)	*televisionstation-affiliatedwith*	broadcastNetwork formerBroadcast-Network	Broadcast-Network(100)
Artist(98.8) Writer(1.2)	*visualartistartform*	field	-

Automated Knowledge Base Creation: The linking and filling approach is the most popular way of knowledge generation [13]. The last few years have witnessed some of the major works in automated information extraction systems and thereby targeting at large scale knowledge base constructions with minimal amount of human supervision. To this end, much work has explored unsupervised bootstrapping for a variety of tasks, including the acquisition of binary relations [3], facts [10], and instances [15]. OIE further focused on approaches that do not need any manually-labeled data [12]. Pujara et. al. [16] have used probabilistic soft logic to detect inconsistencies in knowledge graphs by exploiting dependencies within the graph. Furthermore, some pioneering works have been done by Wang et. al. [21] using statistical inference mechanisms (MCMC). However, our approach is different from these methods since, it exploits the open KBs to discover novel facts on a structured KB.

Future Work and Conclusion

In this work, we present a statistical approach to find accurate analogous properties across NELL and DBPEDIA . In the process we combine our probabilistic instance alignment method with this to generate set of facts. Our approach avoids tweaking of multiple parameters. We exploit the data set to train a simplistic model and use the model to learn threshold value across various NELL properties. Our approach generates highly accurate set of new DBPEDIA facts previously not extracted from the Wikipedia info-boxes. This can serve as an additional set of facts to DBPEDIA and thereby proving essential for any LOD based question-answering system.

We were able to generate approximately 1.6K new facts with direct mapping and approximately 0.5K with inverse mapping. These numbers are low given the fact that we started with 96K NELL triples. However, we hope to generate more triples with REVERB since its fact base is approximately 14Mi. Working with REVERB brings on the additional task of clustering similar properties (e.g. *is wife of*, *was married to*, *is spouse of*). Furthermore, our approach suffers from the manual selection of K_{thres}. We want to devise an automated technique to overcome this selection.

References

[1] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. {DBpedia} – {A} Crystallization Point for the Web of Data. *Journal of Web Semantics*, 7(3), 2009.

[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proc. of the 2008 ACM SIGMOD international conference on Management of data*, 2008.

[3] S. Brin. Extracting patterns and relations from the World Wide Web. In *Proc. of WebDB Workshop at EDBT-98*, 1998.

[4] R. Bunescu and M. Paşca. Using encyclopedic knowledge for named entity disambiguation. In *Proc. of EACL-06*, 2006.

[5] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *Proc. of AAAI*, 2010.

[6] S. Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proc. of EMNLP-CoNLL-07*, 2007.

[7] A. Dutta, C. Meilicke, and S. P. Ponzetto. A probabilistic approach for integrating heterogeneous knowledge sources. In *ESWC*, volume 8465 of *LNCS*, pages 286–301. Springer, 2014.

[8] A. Dutta, M. Niepert, C. Meilicke, and S. P. Ponzetto. Integrating open and closed information extraction : Challenges and first steps. In *Proc. of the ISWC-13 NLP and DBpedia workshop*, 2013.

[9] O. Etzioni. Search needs a shake-up. *Nature*, 476:25–26, 2011.

[10] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in KnowItAll (Preliminary results). In *WWW*, 2004.

[11] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proc. of EMNLP-11*, 2011.

[12] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *Proc. of EMNLP-11*, 2011.

[13] H. Ji and R. Grishman. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1148–1158, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[14] K. Lai and N. Cerpa. Support vs confidence in association rule algorithms. In *OPTIMA*, 2001.

[15] M. Paşca and B. Van Durme. Weakly-supervised acquisition of open-domain classes and class attributes from Web documents and query logs. In *Proc. of ACL-08*, 2008.

[16] J. Pujara, H. Miao, L. Getoor, and W. Cohen. Large-scale knowledge graph identification using psl. In *AAAI Fall Symposium on Semantics for Big Data*, 2013.

[17] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2), 2006.

[18] F. M. Suchanek, S. Abiteboul, and P. Senellart. PARIS: Probabilistic Alignment of Relations, Instances, and Schema. *PVLDB*, 5(3):157–168, 2011.

[19] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM.

[20] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Silk - A Link Discovery Framework for the Web of Data. In *Proc. of LDOW '09*, 2009.

[21] D. Z. Wang, Y. Chen, S. Goldberg, C. Grant, and K. Li. Automatic knowledge base construction using probabilistic extraction, deductive reasoning, and human feedback. In *Proceedings of the Joint Workshop on*, AKBC-WEKEX '12, pages 106–110, 2012.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-121

Towards the Usage of Advanced Behavioral Simulations for Simultaneous Tracking and Activity Recognition

Arsène Fansi T. [a,b] and Vincent Thomas [a] and Olivier Buffet [a] and Fabien Flacher [b] and Alain Dutech [a,1]

[a] *INRIA / Université de Lorraine, Nancy, France*
[b] *Thales Services, Vélizy, France*

Abstract. Tracking and understanding moving pedestrian behaviors is of major concern for a growing number of applications. Classical approaches either consider the two problems separately or treat them simultaneously while relying on limited context-based graphical models. In this paper, we present an approach to tackle both the problems conjointly based on richer contextual information issued from agent-based behavioral simulators which aim to realistically reproduce human behaviors within complex environments. We focus on the special case of a single target and experimentally show that the proposed approach manages to track a single pedestrian with complex behavior even in case of long periods of occlusion.

1. Introduction

The ability of using sensor networks to track and understand the behavior of moving human beings is of great importance for a wide range of applications such as surveillance [1] and smart homes [2]. When sensors are cameras, this implies retrieving behavioral information from image analysis. This is not a trivial task to perform, even for humans, and interpretation errors are common. Moreover, when the considered environment is not fully under sensory coverage, one problem of interest is to determine what is the behavior of the tracked targets while being in a non-covered area.

Understanding a pedestrian's behavior is intimately coupled with tracking its location. Moving pedestrians are usually driven by an inner motivation in relation to the activity they are performing in the environment. Therefore, location and motivation are contextually dependent, and the knowledge of one may help estimate the other. However, there are different levels of granularity one can refer to when seeking to understand people's behavior. Imagine people walking through a given ticket machine for withdrawing cash, we may distinguish, in a hierarchical order, *atomic actions* (e.g., walking), *interaction* (e.g., approaching, queuing, avoiding) and *activities* (e.g., withdrawing cash) [2].

[1]Contact Information: {arsene.fansi-tchango, vincent.thomas, olivier.buffet, alain.dutech}@loria.fr, fabien.flacher@thalesgroup.com

[2]In the literature, the term "*activities*" is often used ambiguously to refer to atomic actions or interactions.

These different levels are clearly not independent and, the higher the level, the more the scene's context has to be taken into account.

Most of the existing works, when not considering the problem of tracking and activity recognition separately, rely on limited context-based graphical models for improving trajectory estimation from the estimated activity and vice-versa. We approve the idea of taking advantage of the scene information by performing both operations simultaneously. Moreover, we argue that considering tools providing richer contextual information is an asset to deal with high-level activity recognition even in more complex scenarios, a feature of particular interest for monitoring systems.

Nowadays, the use of simulators to generate contextual-based realistic human behaviors in indoor environments has become very popular in a wide range of application domains[3,4]. These simulators usually encapsulate a model of the considered building (together with the objects therein); and a description of authorized activities is provided, reflecting the prior knowledge about the context of the environment. Works from the situated artificial intelligence field have focused on designing reactive virtual agent control architectures based on sensorimotor loops [5,6] whose purpose is to define, at any time, the behavior a virtual agent will have according to his internal state and current environment context.

In this paper, we are interested in developing a system capable of inferring the activity of pedestrians in indoor non-crowded scenes based on their trajectory, even when they are not under sensor coverage. The proposed approach leverages richer contextual information from such a simulator, which in turn, is integrated within a particle filter for the analysis of people's behavior.

The remainder of this paper is structured as follows. Section 2 provides an overview of related work. Section 3 briefly introduces mathematical background of particle filters while Section 4 describes the implementation details of our system. Section 5 is dedicated to the experimental evaluation of the approach. Finally, Section 6 identifies some important research lines induced by the described work.

2. Related Work

Tracking pedestrians has been of major concern during the last decades. In complex scenes, occlusions often occur and interrupt tracks identified so far. Maintaining relevant information in such situations is of crucial importance in order to assemble complete tracks or understand the behavior of the concerned targets. Works in the literature [7,8,9] have focused on defining mathematical models based on a set of attractive and repulsive fields representing human motion features. Based on these models, methods [10,11,12] have been developed for better handling short occlusions in human motion prediction. However, they do not attempt to understand the behavior of the different individuals.

On the other hand, methods [13,14] have been recently proposed in the literature for action recognition in which authors mainly rely on image feature extraction and feature classification. However, these works do not benefit from advantages available in considering tracking people's trajectories.

Dealing conjointly with location tracking and behavior understanding presents an advantage as it is possible to exploit the relationship between location and behavior. In [15], Bruce et al. represent the state of the pedestrian by his final goal (in terms of

destination point) together with his location and then use a particle filter for inferring the pedestrian goal with the help of a path planner. A limit of their method is that people's behaviors are assimilated to the environment physical points. Moreover, there is no causality modeling between these points.

Wilson et al. [16] formally introduced the simultaneous tracking and activity recognition (STAR) problem and consider the use of dynamic Bayesian networks (DBNs) to represent causal influences and causality through time among different variables representing their system state. However, the granularity of the modeled activities is limited to whether or not a target is moving. In [17], the authors investigate the use of relational DBNs to represent relationships among interacting targets in the STAR problem. They probabilistically model the dynamics of relations between different targets and formalize a dynamic model that takes into account such relationships. While result improvements are noticeable, the graphical model used fails to encapsulate relevant information regarding target-object interactions.

In this paper, we also address conjointly the problem of tracking and activity recognition. However, unlike previous approaches, we take into account richer contextual information. By contextual information, we refer to environmental information perceived by a pedestrian that may affect his behavior. In order to do this, we rely on process-oriented behavioral models for autonomous agents. The purpose of such advanced behavioral models is to determine, for each agent, the action he should undertake in the environment based on his internal state and current perceptions. Such models, as described in [4], are not only characterized by physical attributes (e.g., position, velocity), but integrate, within themselves, action selector mechanisms often coupled with cognitive control architectures [18,19,5,6] responsible for creating and executing navigational plans which may involve interactions with objects in the environment. The description of the different objects together with the interactions they can offer are provided to the simulator in order to easily match agent's current action into corresponding interactions in the scene. We then integrate such a simulator, as a predictive block, within a particle filter for people's behavior estimation and analysis.

The main contribution of this paper is two-fold: (i) by integrating advanced agent-based behavioral simulations, we add significant contextual information that may be useful in understanding target behavior from image sequences; (ii) as the results will show, considering richer contextual data makes it possible to handle long periods of occlusions, times during which the tracked agent is likely to perform several activities.

In the following sections, we will briefly introduce relevant background regarding particle filters and describe how we leverage autonomous agent based behavioral models for pedestrian behavior tracking. For the latter, we focus on the special case of a single target.

3. Particle Filtering

In the field of state estimation, the Bayesian filtering framework [20] provides a recursive way of computing the belief $Bel_t(\mathbf{x}_t)$ regarding the state $\mathbf{x}_t$ of a dynamical system at time t given the (potentially noisy) partial observation $\mathbf{z}_t$. It is assumed the availability of the prior belief Bel_0 about the system's initial state. A particle filter (PF) [21] is an approximation of the Bayesian filter in which the belief $Bel_t(\mathbf{x}_t)$ at time t is represented

by a set of N_s weighted particles $\mathcal{S}_t = \{\mathbf{x}_t^i, w_t^i\}_{i=1}^{N_s}$ where $\mathbf{x}_t^i$ and w_t^i are respectively the state and the weight of the i^{th} particle. The particle set $\mathcal{S}_t$ is typically constructed from the previous set $\mathcal{S}_{t-1}$ and the current observation $\mathbf{z}_t$ as follows [21]:

- **prediction:** a sample $\mathbf{x}_{t|t-1}^i$ is generated from each sample $\mathbf{x}_{t-1}^i$ of the set $\mathcal{S}_{t-1}$ using a proposal density function $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t)$. Usually, $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t)$ is equals to $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, the system's dynamics.
- **weight assignment:** each predicted sample $\mathbf{x}_{t|t-1}^i$ is assigned an importance weight w_t^i computed based on the the observation model $p(\mathbf{z}_t|\mathbf{x}_t)$ of the system as $w_t^k = w_{t-1}^k . \frac{p(\mathbf{z}_t|\mathbf{x}_{t|t-1}^i) . p(\mathbf{x}_{t|t-1}^i|\mathbf{x}_{t-1}^i)}{q(\mathbf{x}_{t|t-1}^i|\mathbf{x}_{t-1}^i, \mathbf{z}_t)}$. Once computed, the importance weights are normalized.
- **resampling:** it consists in deleting or duplicating particles according to their weights. This is usually done by generating a new set of particles $\{\mathbf{x}_t^j\}_{j=1}^{N_s}$ from an approximate discrete representation of $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ given by

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) \approx \sum_{i=1}^{N_s} w_t^i \delta(\mathbf{x}_t, \mathbf{x}_{t|t-1}^i) \tag{1}$$

  so that $p(\mathbf{x}_t^j = \mathbf{x}_{t|t-1}^i) = w_t^i$. $\delta(.,.)$ is the Dirac delta function. At the end, each resulting particle $\mathbf{x}_t^j$ is assigned a weight $w_t^j = 1/N_s$.

Usually, in order to encourage the state space exploration, the resampling phase is not performed at every time step, but only when the effective size $\widehat{N}_{eff} = \frac{1}{\sum_{i=1}^{N_s}(w_t^i)^2}$ of the filter goes below a given threshold N_T. For more details about PFs, see [21].

4. Agent Based Behavior Tracking

The solution we proposed can be represented by Fig. 1. The simulator is assumed to handle virtual-agent microscopic navigational features and object-agent interactions (e.g., escalators, cash dispensers). It is then used within the prediction step of a particle filtering process to estimate the belief regarding both the aimed location and the activity of the pedestrian. Then, the noisy observation (location of the detected targets obtained after an image analysis process) received from the sensor network is used during the correction step of the filtering. We assume to be aware of areas covered by the sensors. Furthermore, for preserving coherence between the real world and the simulated one, the simulator is fed with changes occurring in the real world such as object states modification (e.g., escalator failures) or exogenous events (e.g., fire alerts). We assumed that the video analysis is performed upstream and it is not part of the discussion in this paper.

In what follows, we describe the different models as required by the Bayesian filter and discuss their implementation. Also, we assume that, although a target may interact with objects within the environment, he can not modify their internal state.

4.1. System Dynamics

Given a behavioral model, a state $\mathbf{x}_t$ of an agent contains attributes that are taken into account within his action-selector mechanism, that is all attributes that may play a role

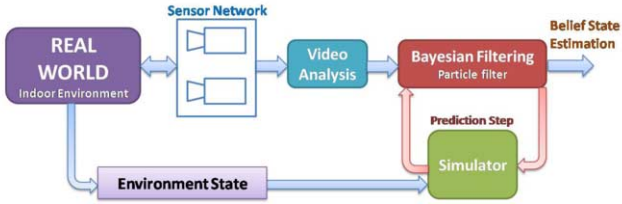


Figure 1. The process overview

in the selection of the actions to be performed by the agent. These attributes can be regrouped into two categories. The first category regroups spatial attributes (e.g., position, velocity) while the second one contains internal attributes representing for example motivations such as psychological and physiological traits (e.g., the thirst level) and resources (e.g., ticket, money) owned. Furthermore, a virtual agent is able to sense his surroundings and builds his proper knowledge of its world (e.g., the object states). This knowledge, combined with its internal variables, represent richer contextual information on which the behavioral model relies for defining self-explanatory agent trajectories; e.g., interrupting his initial plan for getting some drink when thirsty. Therefore, internal attributes may naturally evolve even when the individual is static according to his perceptions.

As the environment state $\mathbf{E}_t$ (including objects) is known and because, as assumed, an agent cannot modify the object states, we only need to consider the agent's dynamics. Since the simulator is in charge of navigational features, the agent's dynamics is fully encoded therein and may be represented as follows:

$$\mathbf{x}_{t+1} \sim f(\mathbf{x}_t, \mathbf{E}_t),$$

where f is the simulator's implemented stochastic function taking as input both the state $\mathbf{x}_t$ of the agent with the environment state $\mathbf{E}_t$ and modifying the agent's inner attributes.

4.2. Observation Model

The observation data $\mathbf{z}_t$ received from the sensors depends on whether the agent is within a covered area or not. However, assuming Gaussian noise with a covariance matrix Q_v, the agent may still be undetected even within covered areas, especially when he is close to the boundaries of both areas. The probability φ of such an event can be approximated as the portion of the Gaussian (represented by a circle of radius r_h) belonging to the non-covered areas (Fig. 2). That is

$$\hat{\varphi} = \frac{\sum \text{prob. of region's cells in non-covered area}}{\sum \text{prob. of all region's cells}}. \tag{2}$$

Finally, we have

$$p(\mathbf{z}_t|\mathbf{x}_t) = \begin{cases} 1 & \text{if } \mathbf{z}_t = \varnothing \text{ and } unc(\mathbf{x}_t), \\ \hat{\varphi} & \text{if } \mathbf{z}_t = \varnothing \text{ and } \neg unc(\mathbf{x}_t), \\ \mathcal{N}_{0;Q_v}(\mathbf{u}_t) & \text{if not,} \end{cases}$$

where $\mathbf{u}_t = \mathbf{z}_t - h(\mathbf{x}_t)$, h is a function extracting the location data from $\mathbf{x}_t$, and $unc(\mathbf{x}_t)$ indicates that the agent ($\mathbf{x}_t$) is within a non-covered area. $\varnothing$ means no-detected agent.

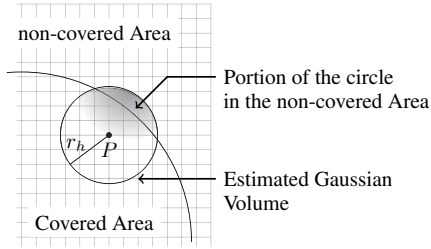


Figure 2. Approximation of φ: P is the agent position. The ratio is computed with respect to all cells in the considered circle.

4.3. Implementation in SE-Star

The simulator used is SE-Star, a Thales proprietary engine capable of modeling adaptive behaviors, navigations and interactions with objects. Each agent is characterized by a motivational tree based on a free flow hierarchy approach [22] containing a set of attributes on which the action selector mechanism relies for computing his current action.

However, SE-Star is a simulator with less random effects in behavior model dynamics and, running several simulations with a given agent state will lead to identical results in terms of behaviors exhibited. Such processes with little noise are not appropriate within a particle filter because of the sample impoverishment phenomenon [21].

Regularized particle filters (RPFs) have been introduced in [23] for preventing this phenomenon. The RPF's idea is to re-sample from a continuous approximation of the probability density function (pdf) $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ instead of its discrete approximation (see Equation 1), hence producing a new particle set with N_s different particles. The continuous approximation of the posterior pdf is computed as follows (see Fig. 3):

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) \approx \hat{p}_\lambda(\mathbf{x}_t|\mathbf{z}_{1:t}) = \sum_{i=1}^{N_s} w_t^i.K_\lambda(\mathbf{x}_t - \mathbf{x}_t^i),$$

where $K_\lambda(\mathbf{x}) = \frac{1}{\lambda^{n_x}}K(\frac{\mathbf{x}}{\lambda})$ is the rescaled kernel density, $K(.)$ is a kernel function, λ is the bandwidth and n_x is the dimension of the state space. We use the Gaussian kernel and, according to [24], the corresponding optimal bandwidth is given by $\lambda_{opt} = \left(\frac{4}{(n_x+2)N_s}\right)^{\frac{1}{n_x+4}}$. Also, when the pdf is multi-modal, it is suggested [24] to have $\lambda = \lambda_{opt}/2$.

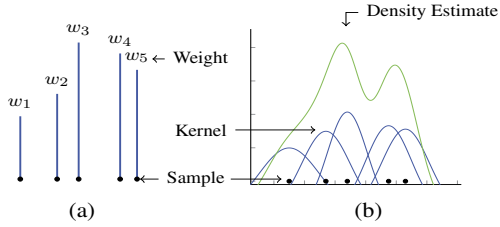


Figure 3. Regularization Process: (a) - Weighted empirical measure; (b) - Regularized measure

[2]Thales Group is a French multinational company that designs and builds electrical systems and provides services for the aerospace, defence, transportation and security markets.

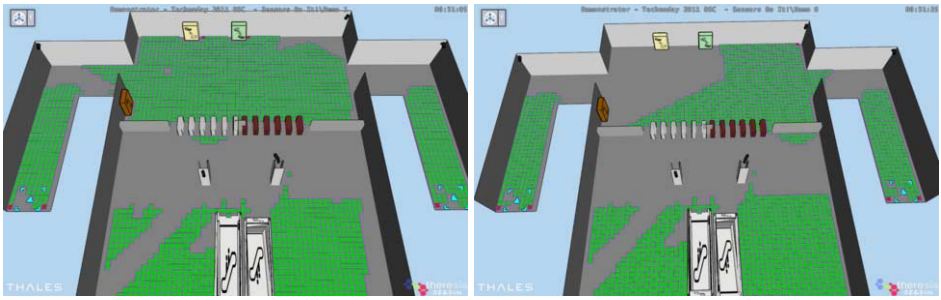


Figure 4. Scenario 1 (left) and 2 (right). Green squares represent areas covered by sensors.

5. Experimental Evaluation

We conducted experiments in a virtual environment representing a two-storey subway station (Fig. 4). The station contains an escalator, a train door, an ATM (in light green), a ticket machine (yellow), a beverage dispenser (brown), ticket barriers (white) and exit barriers (red) and an agent may interact with any of these objects in order to fulfill its objectives. Also, the station is equipped with a camera network set up to not completely cover the environment and configured to report noisy passenger location data.

We consider two scenarios. In the first one, the cameras are set to cover areas occupied by the ATM, the ticket machine and the beverage dispenser. In the second one, these objects are no longer covered by the sensors (Fig. 4) in order to assess the robustness of our approach. A passenger may initially own a certain amount of money and/or a valid ticket and, during his lifetime, may be motivated by three goals or actions: taking a train, drinking or leaving.

Tracking in such an environment is challenging since, depending on the resources owned, which are unknown, a passenger may undertake a sequence of interactions (subgoals) with objects in the scene based on his current motivation, thus affecting his trajectory. For example, a passenger willing to take a train may first buy a ticket if he does not have a valid one. However, buying a ticket will require, if he does not own enough money, to get some cash from the ATM. Furthermore, in non-covered areas, nothing prevents the passenger to switch between motivations and perform several interactions. It is our task to infer, solely from the observed trajectory, the corresponding behavior.

Such a passenger model has been designed in SE-Star in which the three motivations are represented by numerical attributes taking values in $[0, 2]$ and the simulator is in charge of their evolution. Also, the model includes two attributes representing respectively the amount of money and the number of tickets owned.

We run our algorithm with 2000 particles and set r_h to be $0.5m$. The noise standard deviation is set to 0.8, 0.8 and $0.05m$ for the x, y and z coordinates respectively. Initially, a passenger has 30% chances to own a number of tickets (an amount of money) chosen uniformly in the interval $[1, 3]$ ($[1, 5]$) and nothing otherwise. For other attributes, we assume a Gaussian distribution $\mathcal{N}(0.75; 0.5)$.

Next, we refer to behavior inferred from the filter the one exhibited by the majority of the particles within the filter in terms of goal (or subgoal). We consider the **similarity** indicator defined below as a criterion to evaluate the exactness between the passenger real behavior and the inferred one:

Table 1. Results: Similarity, Robustness and Mean Square Errors

	% Non Cov.	Goal Sim.(%)	Goal Rob.(%)	Subgoal Sim.(%)	Subgoal Rob.(%)	General MSE	Goal MSE	SubGoal MSE
Sc. 1	32.7	94.31 (± 6.76)	99.17 (± 0.72)	83.26 (± 7.82)	95.02 (± 6.62)	3.08 (± 2.04)	2.35 (± 1.46)	2.27 (± 1.48)
Sc. 2	67.7	90.25 (± 12.27)	99.46 (± 0.26)	72.41 (± 12.23)	92.20 (± 9.42)	4.29 (± 3.48)	4.78 (± 3.45)	4.49 (± 3.74)

$$Similarity = \frac{\sum_{t=0}^{T} D(g(t), g'(t))}{T},$$

where $g(t)$ and $g'(t)$ are the real and the inferred behavior (goal or subgoal) respectively; $D(.,.)$ returns 1 when the two parameters are equal and 0 otherwise.

Another criterion we consider is the **robustness** which corresponds to the ratio of time the filter contains a valid hypothesis regarding the passenger behavior:

$$Robustness = \frac{\sum_{t=0}^{T} P(g(t))}{T},$$

where $P(g)$ is an indicator of the behavior g's presence in the filter.

Moreover, we compute the mean squared error (MSE) of the location data estimated from the filter with respect to the the position of the passenger. This error is computed according to three such location estimates: general estimate (weighted mean of the location of all the particles), goal-based and subgoal-based estimates (weighted mean of the location of particles whose behavior — goal or subgoal — corresponds to the one exhibited by the filter). Finally, each experimentation has been carried out 10 times and results (mean of all the runs) are reported in Table 1.

For scenario 1, in which the passenger spends 1/3 of the time in non-covered areas, the algorithm has relatively high average scores with low standard deviations, regardless of the criterion. It appears that the filter goal-based similarity and robustness are respectively 94.3% and 99.17%. When focusing on passenger subgoal, these values are respectively 83.26% and 95.02%. This decrease with respect to goal-based values is explained by the fact that, given a motivation, there exists a variety of sequence of interactions, according to the agent internal attributes, for fulfilling it. Moreover, paths in the environment do not help discriminate between possible interactions (e.g., ATM and ticket machine as they are close to each other).

In scenario 2, a degradation of performance can be observed. This is due to the fact the passenger is undetected most of the time. However, despite these significant occlusions (2/3 of the time), the approach is still efficient and keeps good estimates of the target's behavior; the goal and subgoal based similarities being respectively 90.25 and 72.41. An example of the result for the subgoal estimation is depicted in Fig 5.

In both scenarios, the robustness criterion is high. This is of particular interest in case of long occlusions for subsequently recovering the target's behavior when re-observed.

6. Discussion and Future work

In this paper, we address the problem of pedestrian tracking and behavior understanding simultaneously. Unlike previous works in the literature, we take advantage from richer

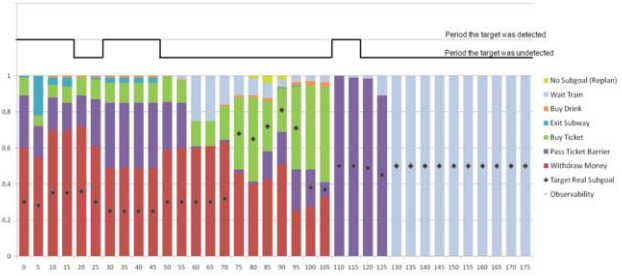


Figure 5. Estimation of the passenger's behavior over a discretized time slot (scenario 2): each bar represents the belief over subgoals (modeled as slices on the bar) by the filter. The black point on each bar represents the actual behavior of the tracked target. The line on top of the bars represents time frames during which the target were either observed or not. We can see that maintaining a valid hypothesis regarding the exact behavior (timesteps 75–105) helps recovering the target's behavior as soon as he is observed again (timestep 110).

contextual information by relying on advanced agent-based behavioral models designed in the field of situated artificial intelligence and corresponding simulators. We integrate within a particle filter such a virtual agent simulator as a predictive block for behavior tracking purposes. We focus on the single target case and evaluate our approach on a virtual environment.

The idea behind evaluating in virtual environments was to perform "witness" experiments in which the behavioral model of real humans is assimilated to a clearly circumscribed model that we perfectly master in such a way to avoid inexplicable phenomena. Indeed, the same behavioral model is used for the real world simulation and within the particle filter in order to verify that, in this extreme case, we actually get a very good prediction. The results detailed in Section 5 show encouraging filtering performances in terms of both behavior (goal/subgoal) and location estimations even in cases of long periods of occlusion.

It would be interesting, in a future work, to confront our system on real scenarios and evaluate the impact of approximating the mechanism of internal decision-making of real humans, which is unknown, by an artificially designed behavioral model. However, when considering the real context, the first step to undertake is the calibration of the real sensor network with the simulator used in the filter. Secondly, we may face the problem of characterizing the duration of real human-object interactions as it may differ from one individual to another given the same object. One solution consists in defining within the simulator, for a given object, a unique duration time representing the average of different interaction durations observed in the real world. Another solution is to include within the artificial behavioral model, an attribute representing the duration of the interaction a given agent is about to perform. The third issue is the realism of the behavioral model with respect to what is generally observed; hence the necessity to work together with experts in behavior analysis for capturing the essence of people's behavior and trajectories within the considered environment. As a very short term objective, we plan to deploy the solution in a Thales office buildings.

The proposed approach presents several advantages. Firstly, by relying on agent-based behavioral models, we clearly emphasize the fact that each individual has its own specificity in terms of reasoning capabilities and strategies. Usually, people do tracking based on simple and stereotyped behaviors (e.g., flow of trajectories). We instead consider finer individual behavior that makes it possible to explain outliers impossible to

explain with stereotyped behaviors. A second advantage is the ability to handle, without any supplementary effort, exogenous events or changes that may occur within the environment (e.g., escalator failure, fire alerts). Indeed, such events, as soon as they are perceived by an individual, are automatically taken into account during the decision process of his behavioral model therefore adapting its behavior accordingly and updating the particles in consequence. Finally, there is no need to build any graphical models (e.g., DBNs) representing relationships between variables of the system state.

As a medium-term objective, we will investigate the multi-target case. This is a very challenging problem as, because of the camera network, the data association problem is added on top of the tracking problem. The data association problem consists in determining from which target a received observation comes. Furthermore, the interactions between targets add another layer of complexity.

References

[1] I. Haritaoglu, D. Harwood, and L. S. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 809–830, 2000.
[2] J. Krumm, "Multi-camera multi-person tracking for easyliving," pp. 3–10, 2000.
[3] S. Stylianou, M. M. Fyrillas, and Y. Chrysanthou, "Scalable pedestrian simulation for virtual cities," in *Proc. of the Symposium on Virtual Reality Software and Technology*, pp. 65–72, 2004.
[4] W. Shao and D. Terzopoulos, "Autonomous pedestrians," *Graph. Models*, vol. 69, pp. 246–274, 2007.
[5] P. Maes, "Modeling adaptive autonomous agents," *Artificial Life*, vol. 1, pp. 135–162, 1994.
[6] J.-A. Meyer, "From natural to artificial life: Biomimetic mechanisms in animat designs," *Robotics and Autonomous Systems*, vol. 22, pp. 3–21, 1997.
[7] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Phys. Rev. E*, vol. 51, pp. 4282–4286, may 1995.
[8] R. C. Arkin, *Behavior-based Robotics*. Cambridge, MA, USA: MIT Press, 1st ed., 1998.
[9] C. Reynolds, "Steering Behaviors for Autonomous Characters," in *Game Developers Conference 1999*, pp. 763–782, 1999.
[10] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *Proc. IEEE 12th Int. Conf. on Computer Vision*, pp. 261–268, 2009.
[11] M. Luber, J. Stork, G. Tipaldi, and K. Arras, "People tracking with human motion predictions from social forces," in *proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 464–469, 2010.
[12] B. Tastan and G. Sukthankar, "Leveraging human behavior models to predict paths in indoor environments," *Pervasive Mobile Computing*, vol. 7, pp. 319–330, jun 2011.
[13] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Proc. of ICCCN '05*, pp. 65–72, 2005.
[14] J. Liu, J. Luo, and M. Shah, "Recognizing realistic actions from videos "in the wild"," 2009.
[15] A. Bruce and G. Gordon, "Better motion prediction for people-tracking," in *Proceedings of ICRA*, 2004.
[16] D. H. Wilson and C. Atkeson, "Simultaneous Tracking and Activity Recognition using many anonymous, binary sensors," in *Proc. of the 3rd Int. conf. on Pervasive Computing*, pp. 62–79, 2005.
[17] C. Manfredotti, D. J. Fleet, H. J. Hamilton, and S. Zilles, "Simultaneous tracking and activity recognition," in *Proc. of ICTAI'11*, (Washington, DC, USA), pp. 189–196, 2011.
[18] X. Tu and D. Terzopoulos, "Artificial fishes: Physics, locomotion, perception, behavior," in *Proc. of SIGGRAPH*, pp. 43–50, 1994.
[19] J. Funge, X. Tu, and D. Terzopoulos, "Cognitive modeling: Knowledge, reasoning and planning for intelligent characters," in *Proc. of SIGGRAPH*, pp. 29–38, 1999.
[20] S. Sarkka, *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.
[21] M. S. Arulampalam, S. Maskell, and N. Gordon, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. on Sig. Processing*, vol. 50, pp. 174–188, 2002.
[22] T. Tyrell, "Defining the action selection problem," 1993.
[23] C. Musso, N. Oudjane, and F. L. Gland, "Improving Regularized Particle Filters," 2001.
[24] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-131

Human Speech Processing for Pedestrian Assistance: Towards Cognitive Error Handling in Spoken Dialogue Systems

Martin HACKER [a]
[a] *Interdisciplinary Center for Embedded Systems (ESI), Department of Computer Science, University of Erlangen-Nuremberg, Germany*

Abstract. Current spoken dialogue systems (SDS) often behave inappropriately as they do not feature the same capabilities to detect speech recognition errors and handle them adequately as is achieved in human conversation. Adopting human abilities to identify perception problems and strategies to recover from them would enable SDS to show more constructive and naturalistic behavior.

We investigated human error detection and error handling strategies within the context of a SDS for pedestrian assistance. The human behavior serves as a model for future algorithms that could yield reduced error rates in speech processing.

The results contribute to a better understanding which knowledge humans employ to build up interpretations from perceived words and establish their confidence in perception and interpretation. The findings provide useful input for SDS developers and enable researchers to estimate the potential benefit of future research avenues.

1. Introduction

Current spoken dialogue systems (SDS) often behave inappropriately when user utterances are incorrectly recognized by automatic speech recognition (ASR). This seems less to be a shortcoming of the acoustic processing but rather a consequence of the fact that SDS fail to detect and overcome these problems. In contrast, humans are able to employ a variety of different knowledge sources to estimate the reliability of hypotheses, interpret partially unreliable fragments and clarify missing or doubtful information during the subsequent course of the dialogue. Thus, to handle errors in the way that humans do would require SDS to implement the following functionalities [1]:

- FUNC1: establish the reliability of an ASR hypothesis and its constituents,
- FUNC2: build possible interpretations based on reliable parts of the hypothesis,
- FUNC3: choose an appropriate dialogue strategy to foster correct understanding, such as the *accept, ignore, clarify, reject* actions suggested in [2].

We believe that the challenge to achieve FUNC3 cannot be overcome without finding a solution for FUNC1/FUNC2. In particular, for the *clarify* strategy it is unclear what part of the transcript should be clarified and how the clarification request should be realized.

The knowledge sources employed by humans for reliability estimation (FUNC1) and interpretation (FUNC2) extend from phonetic and linguistic to situational [3] and common sense knowledge. To replicate human behavior in SDS, it is necessary to understand

which information is utilized by humans in detail, and how this information is integrated during the interpretation process. These issues are subject to research in the fields of neuro- and psycholinguistics. On the other hand, we need to know how this information can be linked back and formalized for a particular SDS domain and whether it can in fact be beneficial for the performance of a SDS.

In this paper, we bridge this gap by considering a given dialogue system for pedestrian assistance and real user utterances collected with this system. By presenting ASR hypotheses in various forms to human subjects, we can activate cognitive processes and investigate human error handling behavior for these utterances. The study design is based on an experimental framework that overcomes methodological difficulties and provides experimental control over a variety of variables (cf. [1]).

The study aims at investigating the following research questions:

1. How well are humans doing in estimating the reliability of ASR hypotheses, depending on the type and amount of information provided by the experimenter? This investigation aims at evaluating which kind of information could contribute to improve error handling in SDS. By investigating the impact individual kinds of features have on the performance in human speech processing (HSP), we can estimate the potential benefit that incorporating such features in ASR may bring and establish if these are promising research avenues to explore.
2. Which criteria are applied to successfully estimate the reliability of hypotheses? How can such criteria be operationalized to improve confidence measures?
3. What strategies are applied by the subjects to interpret utterances that are believed to contain recognition errors? Which constituents and linguistic properties of erroneous transcripts are considered as reliable enough to use them as anchors for an interpretation? The insights are intended to inspire researchers to design novel formalisms for interpreting incomplete syntactical representations, and to be used to inform existing error-corrective mechanisms such as [4].
4. How confident are humans in their interpretations and how does this influence their choice of dialogue strategies? Such insights would help to establish a gold standard for naturalistic clarification strategies.
5. How can the human strategies be replicated algorithmically?

In this paper, we describe the study design and start to tackle some of these research questions. The paper is structured as follows: After summarizing related work and recapitulating the experimental framework, we introduce the dialogue system and the speech corpus the study is based on. Then we describe the experimental setting. We report and discuss the results of our analyses of reliability estimation performance as well as reliability criteria and interpretation strategies and conclude with an outlook on future work.

2. Related Work

Human error handling with respect to perception problems in SDS has been subject to several investigations before. Schlangen and Fernandez [5] simulated perception problems by introducing a noisy channel into human-human conversation. Single words in the audio signal were substituted by noise. The authors investigated which clarification requests were used by the subjects to retrieve the missing information. The results are valuable with regard to how naturalistic clarification requests can be generated. It though

remains unclear, when explicit clarification should be preferred against other dialogue strategies. The choice of the dialogue strategy depends on the subject's interpretation of the corrupted utterance and his/her confidence therein – two variables that were not evaluated in the study. The experimental setting is targeted on non-perception instead of misperceptions that are prevalent in ASR. Applying the observed clarification strategies would require SDS to solve the tasks of FUNC1 and FUNC2 before in order to decide which words should be ignored and on which words the interpretation should be based on. A shortcoming of the use of the auditory channel is that experimenters loose control over the perception of the remaining words since it is unknown whether these words were correctly recognized by the subjects.

Skantze [6] applied a different method that resolves this shortcoming by substituting the noisy channel by an ASR module and visually presenting its output to the subjects that took on the role of the SDS operator. However, the study aimed at evaluating which dialogue stategies facilitated dialogue success after complete non-understanding.

Recent statistical approaches use POMDPs to optimize dialogue policies that allow to recover from misunderstanding [7]. The lack of understanding about why humans applied a particular strategy in the training corpus, however, still causes unnatural behavior.

Skantze and Edlund [8] ran a different experiment to establish a gold standard for ASR word error detection. The subjects were asked to correct ASR hypotheses that were shown together with varying extra information from the recognizer and the dialogue history. The human error detection performance was evaluated in terms of edit operations of correct and incorrect words. The results indicate that humans benefit from both contextual information and information provided by n-best recognition alternatives. As the study includes no qualitative analysis to establish why subjects decided to remove certain words from the transcripts, it remains unclear how the utilized information can be operationalized for their integration into SDS.

3. Study Design

3.1. Framework

For the study, we used the experimental framework described in [1]. The framework provides a method to control and evaluate the influence of different types of information on human speech processing and error handling. It overcomes a methodological difficulty that arises when trying to control the problem source of erroneous ASR transcripts, i.e. incorrect word recognition. Human word recognition is part of the more complex subconscious speech perception and understanding process that experimenters need to split up when trying to gain experimental control at the word recognition stage.

The framework proposes to preassign the results of word recognition by visually presenting ASR hypotheses to the subjects. In order to activate cognitive speech processing, the subjects are instructed to vocalize the given transcript in the head by subvocalization to envision the sound of the utterance without being biased by a concrete acoustic realization. Individual cognitive processes on different levels of perception can be activated and controlled separately by restraining the flow of information of other types:

- *Word recognition* can be controlled by presenting visual stimuli, consisting of single or competing word chains to vary the level of detail for phonetic information.
- *Pragmatic embedding* can be controlled by the amount of contextual information provided to the subjects.

- The *search process* underlying the interpretation of erroneous hypotheses can be controlled by introducing gaps to preassign the results of reliability estimation.

3.2. Dialogue Data

3.2.1. A Dialogue System for Pedestrian Assistance

We used a variant of the pedestrian assistance system ROSE [9] which offers a mixed-initiative spoken language interface. The systems provides the following functionalities:

- calculating routes with public transport connections,
- providing time-table information and live data about delays and cancelations,
- displaying outdoor as well as indoor maps for public transport station buildings,
- giving navigation instructions for pedestrian and public transport routes,
- recommending points of interest (POI) such as shops, cafes or ATMs.

3.2.2. Pedestrian Assistance Corpus (PAC)

For the experiments, we used a subset of a dialogue corpus collected with a Wizard-of-Oz variant of the above described system. The corpus contains utterances of 20 native German speakers (8f / 12m, age 16-68) with some of them speaking strong dialect. The participants were given up to 11 different tasks, resulting in 89 dialogues with an overall number of 544 on-talk user moves having an average length of 5.8 tokens (49% are of length 5 or longer). The tasks are ranging from information retrieval tasks such as public transport live timetable questions to more complex problem solving issues such as navigational assistance, re-planning or recommendation of nearby POI and activities.

The language is characterized by a large amount of named entities denoting public transport stations, line numbers and POI. The domain vocabulary extends to 7351 words 542 of which were actually used in the recordings.

Linguistic Annotation The recordings from the PAC were manually transcribed and annotated. Among other things, colloquial words, clitics, abortions, self-corrections and slips of the tongue were marked and spelling variants were collected. This enables us to ensure a very high quality of our algorithms for alignment and ASR performance evaluation. For example, the phrase “wie viel Minuten sind es bis zu der U-Bahn-Haltestell’ ” and the ASR result “wieviel minuten sinds bis zur u bahn haltestelle” can be matched as identical, whereas standard evaluation algorithms would assign up to 8 word errors.

Speech Recognition and Evaluation For speech recognition we used Google Speech API [10] with open vocabulary and standard language model and, for comparison, Sympalog’s[1] recognition engine SymRec [11] with a bigram language model on the domain vocabulary of 7351 words. The language model was configured with classes for situation-dependent entities such as numbers, stations or POIs. By doing this, we ensured that the language model does not adapt to the concrete tasks used for building up the corpus.

Subset Selection The subset of the corpus that was selected for the study contains 25 utterances for which the ASR transcript contains recognition errors. For 15 of these utterances, we used the output of the Google recognizer, while using the SymRec output for the 10 remaining. The utterances were selected randomly but balanced with respect to length, speaker, underlying task, word error rate (WER) and position within the dialogue.

[1] http://www.sympalog.de

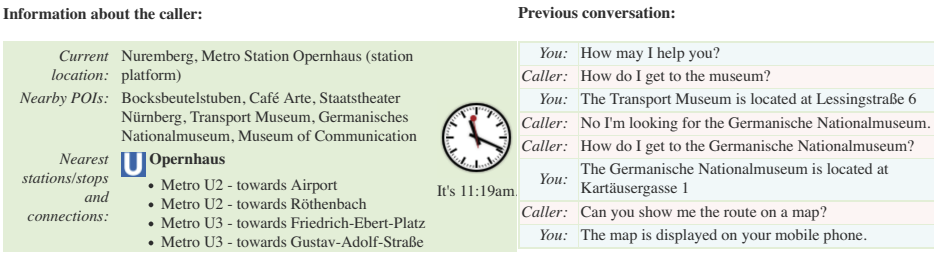


Figure 1. Situational knowledge as provided to the subjects

Context Representation Besides the dialogue history that is implicit in the corpus, the recordings are aligned with the following information representing situational context:

- logical location of the user (i. e. name instead of geographic coordinates),
- nearby points of interest,
- nearby stations and public transport connections therefrom,
- current date and time,
- recommended route if the system did provide one in a previous dialogue step.

Figure 1 shows a sample context representation as has been shown to the subjects.

3.3. Experimental Setting

We decribe the experimental setting of a web-based study based on the above described framework (sec. 3.1) and data (sec. 3.2). The setting has been tested before in a pilot study with 5 participants and accounts for some feedback given by these test participants.

3.3.1. Preparing the Subjects

The participants are instructed to imagine that they work as an operator of a phone hotline, assisting pedestrian customers that call from their mobile phone which makes it sometimes hard to understand what they say. This imaginary context is equivalent to the SDS application described above and is intended to help the subjects understand their task without being required to put themselves into a spoken dialogue system.

Before showing the questionnaires, the subjects are introduced into the information the hotline is intended for (cf. the functionalities of the SDS in section 3.2.1), followed by a thorough instruction about the course of the study and the information that will be provided in the questionnaires. To ensure that the subjects understand the instructions, an example task from another domain is shown (without a given solution to avoid bias).

3.3.2. Tasks

Each task to be done by the participants corresponds to one imaginary phone call and consists of two steps: In a first step, the subject imagines the given context by viewing a representation of the situation (cf. Figure 1) given as one of the variants in Table 1A.

In the second step, a stimulus is presented to the subject. The stimulus consists of a textual or acoustic variant of the ASR hypothesis. The variants used in the study are explained in Table 1B. The user has been instructed before to subvocalize the textual stimuli as proposed in the framework.

Table 1. Possible configurations of the tasks. **A**: context variants, **B**: stimulus variants.

A: ***Context variant***	***Contextual information provided to the subject***
FULL_CONTEXT	complete context as depicted in Figure 1
FULL_DISCOURSE	only discourse history
LAST_DISCOURSE	only the last utterance from the discourse history
FULL_CALLER	only the situational information about the caller
REDUCED_CALLER	reduced situational information about the caller
NO_CONTEXT	no contextual information
MISLEADING	incorrect caller and discourse information

B: ***Stimulus***	***Channel***	***Provided information***
SINGLE	visual	1-best ASR hyothesis
NBEST	visual	5-best ASR list
GAPS	visual	1-best ASR list with misrecognized words substituted by gaps (_____)
AUDIO	auditive	original audio recording with misrecognized words substituted by noise
REFTRANS	visual	transcript without ASR errors
PSEUDO	visual	random word chain generated with the SymRec bigram language model

The presentation of the stimulus is followed by a questionnaire as depicted in Fig. 2. The user is asked to spontaneously interpret the stimulus, to estimate the reliability of the stimulus as perceived and to specify her confidence in the interpretation. The last question aims at evaluating the dialogue strategy the subject would choose as a response.

3.3.3. Participants and Configurations

The experiments were conducted with 36 human subjects (11w, 22m, 3 n/a) with age 19-69, mainly from the academic environment (students and university staff), each performing an individual subset of 9 tasks.

The subset of task configurations and underlying utterances per subject was balanced and rotated among the subjects in order to satisfy empirical standards. In particular, the following conditions are satisfied:

- The intentions of the speakers and the underlying situations (as well as the speakers themselves) are different for all tasks given to a subject.
- For every subject, the set of tasks is balanced with respect to sentence length and word error rate (WER) of the presented hypothesis.

[2]Translation of the German hypothesis: *which opera yard exit oneself take*. The original utterance was: Welchen U-Bahnhof-Ausgang muss ich nehmen? (*Which metro station exit should I take?*)

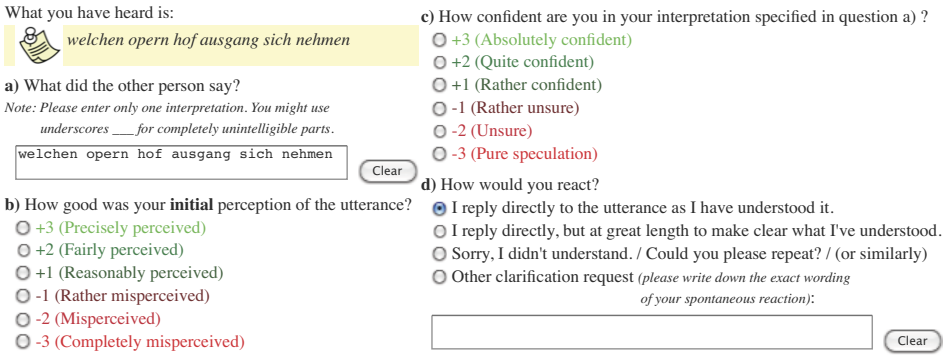


Figure 2. Example questionnaire for a misrecognized[2] utterance corresponding to the situation in Figure 1.

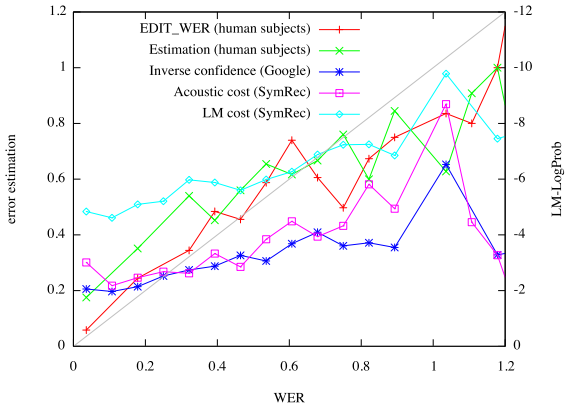


Figure 3. Correlation of the actual WER with human reliability estimation and ASR confidence values.

- For every subject, the task configurations (cf. Table 1) are balanced. Every subject is given 2 of each `SINGLE/NBEST/GAPS/AUDIO` tasks with 3 `FULL_CONTEXT` and 3 `NO_CONTEXT` variants and 2 of the partial context variants, and one additional task with one of the control configurations `REFTRANS`, `PSEUDO` or `MISLEADING`.
- The task ordering is rotated in order to avoid ordering bias.

3.3.4. Qualitative Analyses

At the end of the study, a separate questionnaire is shown where the participants are asked to reflect their answers. The tasks and answered questionnaires are shown again in read-only mode. The subjects are asked to describe what influenced their reliability estimation and what made them choose the specified interpretation.

The participants provided qualitative answers for 212 of the 287 completed tasks. These answers have proven to be extremely valuable during our analyses.

4. Results

4.1. Reliability Estimation

We investigated how well the subjects performed in estimating the correctness of the hypotheses. We excluded the auditory tasks, the tasks with misrecognitions marked (`AUDIO/GAPS`) and the tasks with misleading context from this analysis.

The estimated correctness can be calculated by normalizing the values the subjects used to indicate how well they perceived the utterance. Though the qualitative free text answers where subjects reflected and explained their decisions suggest that many subjects quantified the correctness of an hypothesis rather in terms of interpretability than in terms of word errors. With these observations in mind, we used the word-level edit distance of the subject's interpretation from the original utterance as alternative indication. From the mentioned edit distance, we can calculate a normalized score in a way analogous to word error rate (WER) calculation.

Figure 3 shows the mean correctness estimation for these two alternatives depending on the actual WER. The visualization suggests that the correlation for both alternatives is even higher as for the ASR confidence scores and the language model score that are also included in the figure. It should be mentioned that the human subjects – in contrast to

the speech recognizers – had no access to acoustic information. Hence it can be assumed that the human subjects would perform even better if they could regard such information for their estimation – an assumption that is corroborated by the work of Skantze, who found out that human subjects benefit from ASR confidence information [8].

The human estimations in Figure 3 seem to oscillate compared to the computational measures. This is due to the fact that the latter were computed on the whole PAC corpus which contains about 20 times as many utterances as the subset used for the study[3].

The figure indicates that humans perform better in distinguishing completely correct hypotheses from those containing few errors. In the WER regions above 0.6, humans slightly underestimate the error rate, presumably because they start to be more creative in finding an interpretation.

To summarize, we can state that it is possible to reliably estimate the correctness of speech recognition output without auditory information only on the basis of linguistic and pragmatic knowledge.

4.2. Utilized Information

We used the qualitative answers (see section 3.3.4) to build up a grounded theory of the information utilized for reliability estimation. The resulting classes represent the most important knowledge sources and are listed in Table 2.

With the exception of L_1 and partially L_2, which can be covered by the language model to some degree, as well as A_2, which can be replicated by ASR confidence, the classes refer to knowledge sources that are either not covered by existing computational confidence measures or are considered by separate modules on higher levels of speech understanding without linking the information back to the recognition level. The hu-

[3]In contrast, the computational measures seem to be unreliable for word error rates above 1.0. This can be explained by the fact that there are only 6 utterances with such WER in the corpus while the study subset contained extra tasks of the type `PSEUDO` that show such high WER.

[4]The counts denote the number of cases where a corresponding influence has been mentioned as the most important reason, or has been mentioned at the first place.

Table 2. Classes of information used for reliability estimation.

ID	Description	Count[4]
P	*Pragmatic knowledge:*	
P_1	– Situational, domain and common sense knowledge	41
P_2	– Empathy, knowledge of or personal experience with the presumed speaker intention	9
I	*Interpretability:*	
I_1	– Interpretability based on important words	42
I_2	– Number of possible interpretations (nonambiguous;	11
	too many interpretations; missing contextual information to disambiguate)	11
C	*Completeness:*	
C_1	– Proportion of unreliable or unintelligible (gap tasks) words	15
C_2	– Missing important words	10
L	*Linguistic Knowledge:*	
L_1	– Grammar and syntax	23
L_2	– Combination of syntax and semantics	22
L_3	– Semantic coherence	15
A	*Auditory Information:*	
A_1	– Identification of unreliable words with high phonetic similarity to presumed target words	12
A_2	– Acoustic perceivability (listening tasks); phonetically confusable words (visual tasks)	12

man strategies reported below can inspire researchers to implement a tighter coupling of recognition and understanding modules, as can be found in human speech perception.

It would be of interest to quantify the benefit of the individual knowledge sources. The study design includes variables that enable us to control these sources (cf. Table 1). Our efforts to analyze the data in such a way have revealed that it would be beneficial to collect more data in order to allow for significant findings. We are currently planning to extend the study by recruiting additional subjects and using another subset of the corpus.

4.3. Interpretation Strategies

We analyzed the qualitative answers in which the subjects explained their interpretations. There seems to be a general behavior pattern for which we can find evidence in a large majority of the cases. This pattern can be outlined as the following 10-step strategy:

1. Identify reliable and important key words.
2. Try to capture the basic syntactic structure of the utterance.
3. Build possible interpretations based on these words, guided by situational and empathic expectations (P_1, P_2) or associations with the key words.
4. Try to assign missing information that is necessary to complete the interpretation.
5. If step 4 fails, specify the missing information in terms of syntactic and semantic categories.
6. Identify unreliable words (mainly by employing P, L_3, A_2).
7. Try to replace these words by phonetically similar words, augmented by
   - the syntactic and semantic categories of the missing information,
   - the identified syntactic structure,
   - domain- and situation-specific vocabulary,
   - semantic coherence (associations with or logical relations to reliable words).
8. Delete unreliable words that cannot be substituted.
9. Try to verify the resulting hypothesis by
   - trying to form a complete sentence with further substitutions or insertions of function words,
   - re-assessing the plausibility of the utterance considering all details,
10. Establish the confidence in the interpretation by regarding its plausibility and the number of alternative plausible interpretations.

Step 1 and step 4 are related to two basic spoken language understanding technologies used in many SDS: *keyword/keyphrase spotting* and *slot filling*. Although these mechanisms are often regarded as "unintelligent", they seem to be cognitively adequate in some respects, particularly when they are combined with rich confidence estimation. Unlike SDS, humans supplement these strategies with several verification and grounding steps in which information is passed back to the lower levels of processing. Instead of simply ignoring the segments not covered by key phrases, humans still try to recognize the correct words augmented by the information gained in the higher levels. Only if this grounding is successful, the interpretation is considered as highly reliable.

We should be aware that human speech processing is a highly interactive process where information is exchanged in any direction at any time. This complex process cannot be reduced to a plain script as outlined above. Our analyses though suggest that the script might reflect the most important flow of information with respect to speech processing in a limited domain with comparatively simple user statements.

5. Summary

We presented a study that aimed at investigating human speech processing of incorrectly recognized utterances as a model for error handling in spoken dialogue systems. In this paper, we reported our analyses how well the subjects performed in estimating the correctness of the given speech recognition hypotheses, what kind of information their estimations rely on and what strategies they apply in order to interpret the utterances.

We state that it is possible for humans to reliably estimate the correctness of speech recognition output only by utilizing non-acoustic information. The analyses of reliability criteria can be a first step towards developing confidence measures that integrate rich information from various knowledge sources and levels of processing.

The strategies humans applied to interpret the corrupted hypotheses disclose a way how future automatic speech understanding technologies might be designed. A cognitively adequate approach would combine the typical keyphrase spotting strategy with a kind of "grounded slot filling".

There is a number of open research questions that need to be investigated before the human behavior can be replicated in dialogue systems. As a next step, we will evaluate how well the subjects performed in correcting the hypotheses. Further effort will be put into the investigation of the dialogue and clarification strategies used by the subjects in response to the interpretation task.

Acknowledgements We would like to thank the Embedded Systems Initiative (ESI, `http://www.esi-anwendungszentrum.de`) for funding and David Elsweiler for his valuable advice in various empirical issues.

References

[1] Martin Hacker, David Elsweiler, and Bernd Ludwig. Investigating human speech processing as a model for spoken dialogue systems: An experimental framework. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, pages 1137–1138. IOS Press, 2010.

[2] Malte Gabsdil and Oliver Lemon. Combining acoustic and pragmatic features to predict recognition performance in spoken dialogue systems. In *Proceedings of 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, pages 344–351. ACL, 2004.

[3] Martin Hacker. Context-aware speech recognition in a robot navigation scenario. In *Proc. of 2nd Workshop on Context Aware Intelligent Assistance (CAIA 2011)*, pages 4–17. CEUR-WS.org, 2011.

[4] Bernd Ludwig and Martin Hacker. Why is this wrong? – Diagnosing erroneous speech recognizer output with a two phase parser. In *Proceedings of ECAI 2008*, pages 323–327. IOS Press, 2008.

[5] D. Schlangen and R. Fernández. Speaking through a noisy channel: experiments on inducing clarification behaviour in human-human dialogue. In *Proc. Interspeech 2007*, pages 1266–1269. ISCA, 2007.

[6] Gabriel Skantze. Exploring human error recovery strategies: Implications for spoken dialogue systems. *Speech Communication*, 45(3):325–341, 2005.

[7] Jason D. Williams and Steve Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007.

[8] Gabriel Skantze and Jens Edlund. Early error detection on word level. In *COST278 and ISCA Tutorial and Research Workshop (ITRW) on Robustness Issues in Conversational Interaction*, 2004.

[9] Bernd Ludwig, Bjørn Zenker, and Jan Schrader. Recommendation of personalized routes with public transport connections. *Intell. Interactive Assistance and Mobile Multimedia Comp.*, pages 97–107, 2009.

[10] Mike Schuster. Speech recognition for mobile devices at Google. In *Proceedings of the 11th Pacific Rim International Conference on Trends in Artificial Intelligence*, PRICAI'10, pages 8–10. Springer, 2010.

[11] Elmar Nöth, Axel Horndasch, Florian Gallwitz, and Jürgen Haas. Experiences with commercial telephone-based dialogue systems. *it–Information Technology*, 46(6/2004):315–321, 2004.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-141

A! – A Cooperative Heuristic Search Algorithm

Antti HALME [1]
Aalto University, Finland

Abstract. We propose a new parallel search algorithm – A! – based on cooperating A* search agents, concurrency and a secondary tiebreaking heuristic. The search agents in A! share information asynchronously and trade some of their independence for additional search focus and a more global view of the search task. A! is inherently nondeterministic due to the implicit randomness of instruction scheduling, but given a consistent primary heuristic, it still finds optimal solutions for the single-source shortest path problem (SSSP). A! combines into a single cooperative search algorithm the breadth available in parallel execution and the depth-first orientation of both locally and globally informed search.

We experimentally show that A! outperforms both vanilla A* and an explicitly randomized, noncooperative parallel A* variant. We present an empirical study on cooperation benefits and scalability in the classic 15-puzzle context. The results imply that cooperation and concurrency can successfully be harnessed in algorithm design, inviting further inquiry into algorithms of this kind.

Keywords. A*, heuristic search, parallel algorithm, cooperation, nondeterminism

1. Introduction

Search features in numerous real world applications from pathfinding to resource optimization. Application of heuristic information can improve searching performance by providing focus for search space exploration. While heuristics distinguish promising areas of the search space from those less likely to lead to progress, there still remains plenty of searching to do, as there typically exist multiple equally good directions to pursue.

For the single-source shortest path problem (SSSP) the standard heuristic search method is the A* algorithm [1]. A* follows a best-first strategy that considers both known distance from the start node and an estimate for the remaining distance to a goal node. As a graph-based algorithm, A* is fundamentally hard to parallelize: best search directions depend on overall search progress. Nonetheless, there has been some success in parallelizing best-first search both locally and in a distributed fashion, mostly through insightful search space partitioning and redundancy elimination [2–6].

In this work we explore parallel heuristic search that is based on cooperation and dependence rather than on work division and independent execution. We focus on the cooperation of multiple distinct worker components, algorithmic agents, executing con-

[1] Address: Antti Halme, Aalto University, Dept. of Information and Comp. Sci., PO Box 15400, FI-00076 Aalto, Finland; E-mail: antti.halme@aalto.fi; Web: http://users.ics.aalto.fi/ahalme/coop/ .

currently and communicating asynchronously. Our main hypothesis is that cooperating agents searching together can be more effective than agents searching in isolation. We test our hypothesis in a series of computational experiments and look for some general trends. This paper summarizes the research presented in full as a master's thesis [7].

The contribution of this paper is a new kind of cooperative heuristic search algorithm, a parallel variant of A*, dubbed A!. A! features cooperating search agents that share information and collectively maintain a secondary tiebreaking heuristic. Agents in A! explore the search space in an implicitly random fashion, directed by this dynamic ranking heuristic. Indeed, we view cooperation simply as another layer of heuristic focus, a depth-first orienting sense of global search progress.

We empirically evaluate A! by solving instances of the standard 15-puzzle benchmark problem. The performance of A! is compared with vanilla A* and a randomized non-cooperative parallel A* variant. We show that A! outperforms both methods. The implementation is not as such competitive against state-of-the-art parallel 15-puzzle solvers, but rather a first step in applying cooperation as a secondary heuristic.

We begin with a discussion on cooperative search in the next section and then describe the A! algorithm. Results from computational experiments on 15-puzzles are discussed in Section 4. Related work is discussed briefly before the conclusion in Section 5.

2. Constructing cooperative search

Kishimoto et al. [4] note that combining heuristic search and parallel processing is challenging because of three kinds of overheads. *Search overhead* occurs when the parallel version expands more nodes than the sequential one, typically as a result of non-disjoint search space division between search agents. *Synchronization overhead* refers to the idle time wasted at synchronization points, such as locks on shared data. Finally, *communication overhead* occurs whenever information is exchanged.

Parallel search is usually based on search space division, the goal being the removal of search overhead [4]. This renders synchronization and communication the bottlenecks. The parallelism in heuristic search algorithms is therefore typically very simple, with search agents mostly working independently. While this approach is common in general parallel computing as well, it does not fully leverage the parallel potential of modern multi-core shared-memory systems.

In this work we focus on removing the synchronization overhead and also give up precise communication patterns. Instead of maintaining a clear separation between search agents, we emphasize their close cooperation. Specifically, we make use of asynchronous message passing and nondeterministic instruction scheduling, and build a cooperation mechanism that is powered by the indeterminacy inherent in concurrency. We explore computation that is *implicitly random* as opposed to being explicitly randomized.

The idea in cooperative search is to direct and focus the search by sharing information among concurrently operating search workers [8, 9]. The goal is to search faster as a collective, with each agent utilizing the information they obtain. The hypothesis underlying this work is that cooperating agents outperform agents in isolation: search agents make good use of shared information and the overall effort benefits from cooperation.

Fundamentally, cooperation is communication: cooperating agents consume and contribute messages, whereas isolated agents keep to themselves. Parallel search is by

default concurrent when the cooperation between the agents is unconstrained and asynchronous. Instead of enforcing determinism and working against the natural disorder of concurrency, in this work we embrace it and try to use it to our advantage.

We wish to have a cooperation mechanism that enables information sharing among concurrently operating search workers and is both effective and lightweight enough for this extra effort to be worth it. We wish to augment heuristic search in an unobtrusive way with a basic cooperation mechanism and set the stage for concurrent phenomena.

Instead of a rigid master-slave approach, we want the overall global search to be managed collegially by several processes. We do away with superfluous synchronization and rather embrace asynchronous interaction. A* is point-initialized by nature, so having a portfolio of different approaches would make sense, but to study cooperation specifically, we stick to a uniform strategy. In the terminology of Crainic and Toulouse [8], our full cooperation policy can then be classified as $pC/C/SPSS$.

3. The A! algorithm

3.1. Overview

The A! (*a-bang*) algorithm is a parallel best-first heuristic search that employs asynchronously communicating software agents as concurrently cooperating search workers. Given a graph, a start node, a goal predicate and *two* heuristic functions – a primary and secondary one, denoted $h(u)$ and $\hat{h}(u,v)$, for all u and v in the graph – A! finds a single pair shortest path from the start node to a goal node.

A! consists of N agents that each run a distinct upgraded version of A* search. Each agent performs a heuristic search starting from the start node, but also participates in the cooperation effort: the agents share information about their progress with their fellow agents. An additional message broker entity can be used to streamline communications.

The primary heuristic is the one used in an A*-like graph search itself, while the secondary heuristic serves as a tiebreaker between equally good next-to-open candidate nodes. Vanilla A* simply maintains an estimated value priority queue, but A! workers aim to discern differences between nodes valued equally interesting in the queue.

This is the crux of A!: where vanilla A* always selects the head of the estimated value priority queue, A! agents choose among up to k most promising nodes of equal value based on information they individually acquire during the search.

As information is diffused asynchronously, A! workers have varying notions of search progress. When the priority queues and candidate sets have some differences, the agents diverge, but directed by the heuristics, they also meander close together again. A! generates diversity from concurrency, but also maintains cohesion through heuristics.

For the information to share, best encountered nodes are a simple, effective choice. This directly implies a *distance-to-best*-based secondary heuristic, where each worker is directed towards the areas of the search space that have been fruitful in the past. More elaborate information sharing and utilization schemes are well worth exploring in applications, but lie outside the main focus of this work.

Note that the degenerate case of a single A! agent is *not* necessarily vanilla A*. With only a single agent passing progress information to the secondary heuristic function, the search turns into a momentum-based eager search, where candidates close to recently

opened ones are ranked high among nodes that are equal with respect to the primary heuristic. This *momentum effect* is visible later on in the experiments.

Finally, A! retains the optimality of A* in the sense that the solution paths are the shortest possible, if the primary heuristic is consistent. The secondary heuristic only selects the order in which areas of the search space get explored. The primary and secondary heuristics can be the same distance measure, but this is not necessary in general.

3.2. Algorithm details

Next we present the entire A! algorithm as a collection of pseudocode snippets. We begin with Algorithm 1, which serves as the main body of the algorithm. In short, we simply launch a collection of search workers and wait for one of them to finish. We encapsulate the cooperation communication into a message broker entity that takes care of the communication scheme following a publish-subscribe pattern.

Given the problem instance, including the *two* heuristic functions, the algorithm returns a path from the start node to the found goal node, if one is reached. The path is derived from a path map of successor nodes by the first worker to find a goal. Solution discovery triggers main program termination and solution path retrieval via `getPath(..)`.

The workers run `A!Solver`, outlined in Algorithm 2, which has four parts inside a loop that is repeated until program termination. The first part, lines 6–7, is the node visit, where we check whether the current node is a goal based on the `isGoal` predicate. If it is, we derive the solution path and terminate, if not, we mark the node visited.

The second part, lines 8–14, is the A* expansion. We use the primary heuristic function to estimate remaining distances for the legal neighbors of the current state and update the data structures as we discover new nodes. The priority queue `openHeap` maintains the unopened node queue ordered by estimated total cost. The `pathMap` maps nodes to one another, establishing the successor relation used in solution path derivation.

The third part, lines 15–16, is a cursory peek into the up-to-date `openHeap`. In A! we examine some interesting nodes and select among them according to the secondary heuristic, whereas in vanilla A* we simply draw one node from the top. The peek is a bounded traversal of the priority queue, where we build a list – the `peekList` – of nodes with a cost equal to the top node. If there are no nodes left, the search terminates.

The final part, lines 17–18, contains the selection routine, which for A! is given as Algorithm 3. After one of the nodes has been selected – in one way or another – it is removed from `openHeap` and turned into `current`. Removing nodes is a relatively fast operation for some priority queue implementations, including the Fibonacci heap.

Algorithm 1 : **A!Search**

Require: $N > 0$, NODE *start*, PREDICATE *isGoal*, HEURISTIC h, $\hat{h}$
Ensure: *path* from *start* to nearest node satisfying *isGoal* is shortest possible
1: $mb \leftarrow MsgBroker()$
2: **for** $i = 0$ to N **do**
3: $\quad workers[i] \leftarrow A!Solver(mb.portOut, mb.portIn, s, isGoal, h, \hat{h})$
4: **end for**
5: **for each** *worker* in *workers* **in parallel** *worker.launch*() **end for**
6: **wait** for termination
7: **return** $path \leftarrow getPath(workers)$

Algorithm 2 : **A!Solver**

Require: PORT *portIn*, *portOut*, NODE *start*, PREDICATE *isGoal*, HEURISTIC h, $\hat{h}$

```
 1: openHeap ← FibonacciHeap⟨INTEGER, NODE⟩
 2: closedSet ← Set⟨NODE⟩
 3: pathMap ← Map⟨NODE, NODE⟩
 4: current ← start
 5: repeat
 6:    if isGoal(current) then terminate(current, start, pathMap) end if
 7:    closedSet.add(current)
 8:    for each n in current.getNeighbors() do
 9:       if closedSet.contains(n) then continue end if
10:       g ← current.g + dist(current, n)
11:       f ← g + h(n)
12:       improved ← openHeap.update(n, f)
13:       if improved then pathMap.update(n, current) end if
14:    end for
15:    peekList ← openHeap.getPeekList()
16:    if isEmpty(peekList) then terminate() end if
17:    current ← A!Select(peekList, portIn, portOut, h, ĥ)
18:    openHeap.remove(current)
19: until termination
```

The selection routine `A!Select` is the real core of the A! algorithm: it contains the cooperation functionality and the application of the secondary heuristic. In the first part, lines 1–2, the cooperation routine `asyncRecv` brings new information into the agent. The read is asynchronous and nonblocking in that if there is nothing to receive, the algorithm proceeds without any delay. The routine in Algorithm 3 gives a version with best-information being shared, but other schemes can naturally be constructed here.

The second part features the inclusion of the new information, lines 3–8, as encapsulated into the secondary heuristic function, $\hat{h}$. The `peekList` is sorted on $\hat{h}$ and the highest ranking node is selected. The third part, lines 9–12, mirrors the first part: new data is sent for others to process. The listing shows a small communication overhead optimization, where the agent only informs others, if it believes that it has made progress.

Alternative selection policies to `A!Select` include the random (A?) and vanilla (A*) selection routines. The former selects nodes at random from the peeklist, while the latter simply takes the head of the list. We proceed with an evaluation of these selection policies in a series of computational experiments.

4. Solving 15-puzzles with A!

In this section we describe experiments based on repeated executions of three versions of heuristic search, all based on a single implementation: vanilla A*, a randomized parallel variant A?, and the cooperative A!. We focus on two dimensions that are especially interesting from a cooperative search point of view: the overall benefit from cooperation and the extent to which the methods are scalable. We first describe the experimental setting and then present the computational results. More results can be found in [7].

Algorithm 3 : **A!Select**

Require: LIST *peekList*, PORT *portIn*, *portOut*, HEURISTIC h, $\hat{h}$
Ensure: *select* is the most promising node in *peekList* according to $\hat{h}$ on *best*

```
1:  update, updateH ← asyncRecv(portIn)
2:  if updateH < bestH then best, bestH ← update, updateH end if
3:  select ← peekList.pop()
4:  selectD ← ĥ(select, best)
5:  for each node in peekList do
6:      d ← ĥ(node, best)
7:      if d < selectD then select, selectD ← node, d end if
8:  end for
9:  if h(select) < bestH then
10:     best, bestH ← select, selectH
11:     asyncSend(portOut, {best, bestH})
12: end if
13: return select
```

4.1. Experimental setting

The implementation used in the experiments is based on an A* for n-puzzles implementation by Brian Borowski[2] (BBI). The Java program features an A* solver as well as a version of *IDA**, of which the former was extended to a cooperative version in this work. All tests were executed on a cluster comprising a mixture of blade servers featuring 2.6GHz Opteron 2435, 2.67GHz Xeon X5650, and 2.8GHz Xeon E5 2680 v2 processors.

Two standard heuristics were used in the experiments: Manhattan distance with linear collisions (LC) [10], and a 6-6-3-partitioned static disjoint pattern database for the 15-puzzle (PDB) [11]. PDB was chosen as the primary heuristic and LC-to-best as the secondary heuristic: we use the database to focus on the right areas of the search space and break ties between equal valued nodes by LC-distance to the best observed node.

The test suite is a randomly generated collection of 15-puzzle instances. Instances were solved with BBI and grouped by the length of the optimal solution path, the shortest sequence of moves from the start position to the goal position. Randomly generated impossible instances – off-parity with respect to the target goal state – were discarded.

The instances within a given optimal length group proved to differ greatly in their difficulty: the average number of nodes examined during the search for instances in any group covers a range of several orders of magnitude. Further, as the methods under evaluation have stochastic and nondeterministic properties, runs on a given instance are themselves subject to nontrivial variation. The grouping is still justified, as with enough instance in each group, some general trends become apparent.

The experiments focused on *the number of nodes opened by the winning agent*, which was found to be a reasonably good metric. The goal in this work was not to build a competitive 15-puzzle solver, but to study cooperation effects in A!, so runtime is not considered here. Still, the winning agent measure reflects both the total work done by *all* agents – including repetition – and, to some extent, the total runtime as well. This follows from agents exploring states at roughly the same rate given a core per agent.

[2] http://www.brian-borowski.com/Software/Puzzle/

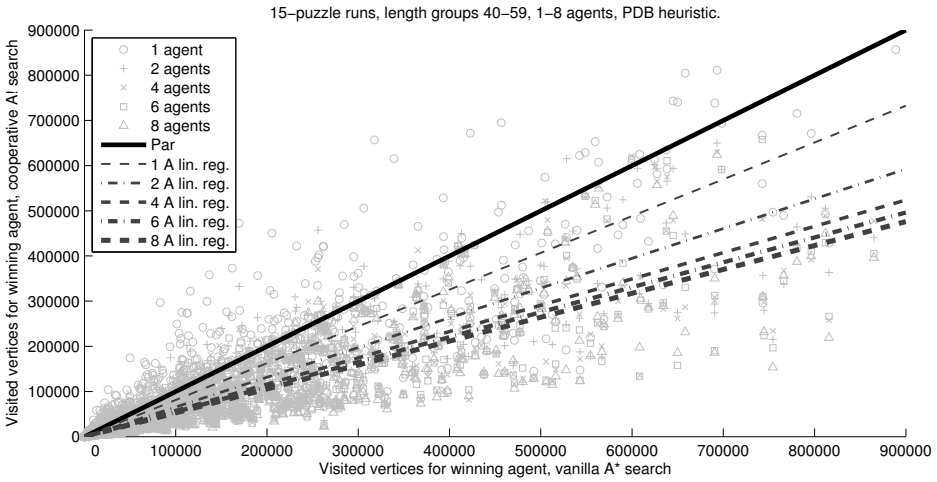


Figure 1. Relative performance of A* (x-axis) and A! (y-axis). The solid line is par, so data points below it represent instances for which A! performs better than A*. Agent count is evaluated in five batches – 1, 2, 4, 6, and 8 agents – with the respective trend lines showing how the configurations compare. With trend lines approaching $\frac{1}{2}$, the winning agents in multi-agent A! open roughly half the states of a vanilla A* run.

4.2. Computational results

The experiments give an overview of the performance of A! in comparison with vanilla A* and a non-cooperative random selection variant A?. We first show that A!, featuring cooperation and the secondary heuristic based ranking, overcomes both vanilla A* and A?. Second, we show that while the returns are diminishing, having more agents improves the overall performance of both A? and A!, but that A! clearly outperforms A?, making a preliminary case in favor of cooperation.

To see how A! fares against the competition, we set the algorithms to solve a suite of instances and observed how many search graph vertices the winning agents open. Figure 1 shows 100 15-puzzles from each of the 40–59 optimal path length groups run on A* and A! in five agent configurations: 1, 2, 4, 6, and 8 agents. We use the median of five runs: this was found to be a reasonable compromise between result quality and available computing resources.

We see the majority of the points falling under the solid par-line, indicating that search using A* is more sluggish than with A!: more states get visited before the optimal solution path is found. Some instances above the par-line – especially for the one agent case – show the vanilla algorithm outperforming A!, likely due to the secondary heuristic being misinformed about the best direction. This is the cost from going depth-first over breadth: all heuristics can be fooled. The trend lines still validate A!. With eight agents, the slope approaches $\frac{1}{2}$, indicating that on average A! needs to see only half the states as vanilla A*. Similar results for A! vs. A? are found in [7].

Searching efficiently in parallel requires a scalable algorithm. Figure 2 shows that A! outperforms A* and A?, and that adding new agents to A! increases its performance. The figure shows runs in optimal length groups, with means of the 100 instances in each group forming a trend line for each of the agent configurations. The group trends are normalized with respect to vanilla A* performance. Finally, the trend lines themselves

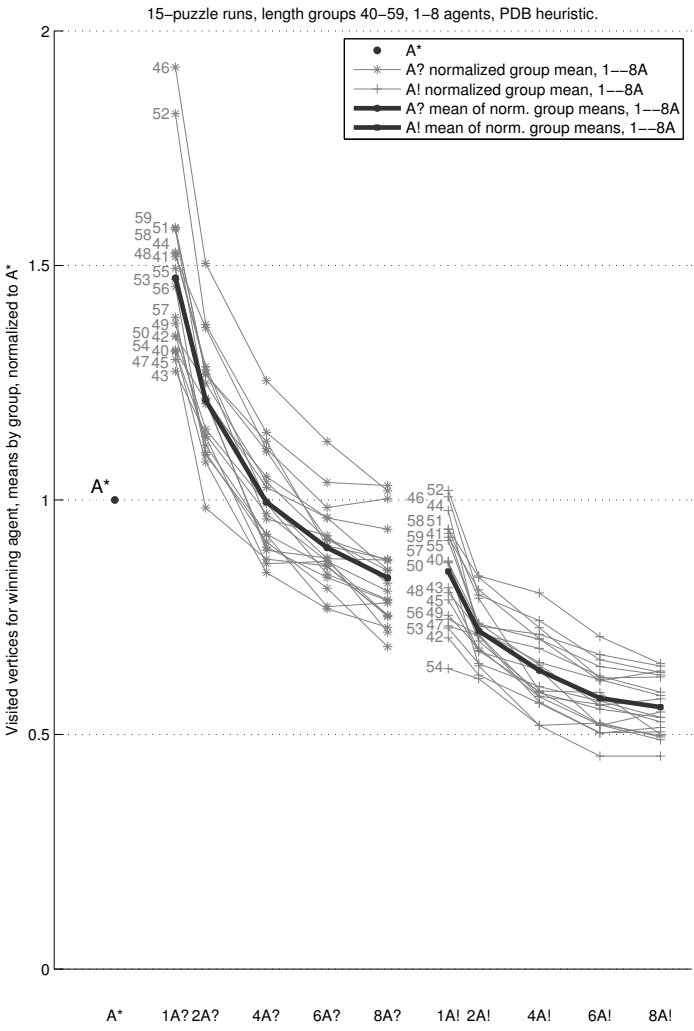


Figure 2. A*-normalized length group means demonstrating scaling benefit from adding more agents. Each line represents a set of instances run over several configurations and repeats, for each method and with the performance scaled to the vanilla A* case. We can see the relative benefit of adding more agents, but also diminishing returns for each expansion. The trend lines overlap to a moderate extent – further illustrated by the mean – suggesting rudimentary asymptotic bounds for the approaches.

are averaged over for a pair of thick mean-of-means curves that summarize over the thousands of data points drawn evenly from the $40-59$ optimal path length range.

We see that A?, perhaps in the spirit of *random restarting* (repeated local search), initially performs worse than A*, as the random selection policy is inferior to a systematic approach, but then with more agents the probabilities turn in its favor. In contrast, A! already starts off well, due to the momentum effect, and gains more power as more agents begin to cooperate.

For some of the groups in Figure 2, A! gets close to the $\frac{1}{2}$ threshold in A*-relative expansion, which also appears to be a plateau for general scalability with regards to

this implementation if not the approach itself. While individual groups exhibit erratic behavior, the overall trend is quite clear: A! outperforms A* and A?, and scales to at least a few agents, but with diminishing returns.

5. Related work

Cooperation has been an theme in AI research for years [12, 13]. The taxonomy work of Crainic and Toulouse [8, 14] is a good place to start exploring the literature on cooperative search. Alba et al. [15] offer extensive surveys on a closely related field of parallel metaheuristics. A! is perhaps best categorized as *distributed search* [5], but the approach is motivated by a *multi-agent system* view of algorithmic cooperation [7].

Barbucha [16] and Ouelhadj and Petrovic [9] present ideas that are similar to A!, but feature cooperation more in terms of traditional parallelism. In A! we propose viewing cooperation as a dynamic heuristic, and present unconstrained concurrency and close interaction as a source of search diversity and performance.

Classical planning tasks make good benchmarks and are often featured in parallel A* papers to give the methods credibility beyond puzzles [2, 4]. However, in these contexts, cooperation effects are rarely studied directly. Information exchange, mostly considered from a search space partitioning and distributed load balancing angle, appears to not have been considered from the heuristic point of view taken in this work. Most A* parallelizations are deterministic and the rest explicitly randomized.

Without search space partitioning, parallelism can be achieved in search through parallelizing node processing in a heavy graph, as in the chess machine *Deep Blue* [17]. Algorithm portfolios and hybrids are another easy way to exploit parallelism, an example being the ManySAT solver [18]. Machine learning methods have also been successfully applied to the discovery of parallel configurations for search [19]. Load balancing through duplicate detection [2], hashing [3, 4], or transposition tables [5] might well be useful in improving exploration diversity also in A!.

6. Conclusion

The 15-puzzle experiments show that the cooperative A! algorithm outperforms both vanilla A* and the non-cooperative random parallel variant A? in this context. Adding more agents to A! clearly improves performance, but the returns are diminishing. Search overhead – the lack of explored path diversity – appears to be a limiting factor in A! performance and an issue worth addressing in future work.

A* expands the search broadly in all directions and in an orderly fashion, in a sense being forgetful about search history. A!, in contrast, prefers depth and emphasizes progress and search momentum. A? is forgetful as well, but through *explicit* randomization, the agents can stumble on the right path faster than in systematic browsing – given enough agents. A! embraces concurrency and *implicit* randomization: the parallel agents cooperate in a nondeterministic way in focusing the search effort in areas that have been found promising. The secondary heuristic serves as a global compass that augments the search when the primary heuristic fails to disambiguate between candidates.

A! combines into a single approach multiple unfinished algorithmic ideas: concurrent execution, asynchronous interaction, implicit randomization, globally informed

depth-first orientation, and the use of a secondary heuristic. A far more detailed study is needed for explicating the exact contribution of each of these factors in A! performance. The approach should also be validated in other contexts.

Still, the experiments indicate that nondeterministic cooperation emerging from asynchronous message exchange *can* be beneficial in heuristic search. Next steps could also include combining A! and cooperation ideas with other parallel A* techniques.

Acknowledgements

I am very grateful to Pekka Orponen for supporting and funding this work. I also thank the reviewers for insightful comments. The computational experiments in this work were performed using the computer resources provided by the Aalto University School of Science *Science-IT* project.

References

[1] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[2] Ethan Burns, Sofia Lemons, Wheeler Ruml, and Rong Zhou. Best-First Heuristic Search for Multicore Machines. *Journal of Artificial Intelligence Research*, 39:689–743, 2010.

[3] Matthew Evett, James Hendler, Ambuj Mahanti, and Dana Nau. PRA*: Massively Parallel Heuristic Search. *Journal of Parallel and Distributed Computing*, 25(2):133–143, 1995.

[4] Akihiro Kishimoto, Alex Fukunaga, and Adi Botea. Evaluation of a Simple, Scalable, Parallel Best-First Search Strategy. *Artificial Intelligence*, 195(0):222–248, February 2013.

[5] John W. Romein, Aske Plaat, Henri E. Bal, and Jonathan Schaeffer. Transposition Table Driven Work Scheduling in Distributed Search. In *Proc. of the 15th Nat. Conf. A. I. (AAAI '99)*, pages 725–731, 1999.

[6] Rong Zhou and Eric A. Hansen. Structured Duplicate Detection in External-Memory Graph Search. In *Proc. of the 19th National Conference on Artificial Intelligence (AAAI '04)*, pages 683–688, 2004.

[7] Antti Halme. *Cooperative Heuristic Search with Software Agents*. Master's thesis, Aalto University, 2014.

[8] Teodor G. Crainic and Michel Toulouse. Explicit and Emergent Cooperation Schemes for Search Algorithms. In *Proc. of the 2nd Intl. Conf. on Learning and Intel. Optim. (LION '07)*, pages 95–109, 2008.

[9] Djamila Ouelhadj and Sanja Petrovic. A Cooperative Hyper-Heuristic Search Framework. *Journal of Heuristics*, 16(6):835–857, December 2009.

[10] Othar Hansson, Andrew Mayer, and Moti Yung. Criticizing Solutions to Relaxed Models Yields Powerful Admissible Heuristics. *Information Sciences*, 63(3):207–227, September 1992.

[11] Ariel Felner, Richard E. Korf, and Sarit Hanan. Additive Pattern Database Heuristics. *Journal of Artificial Intelligence Research*, 22:279–318, 2004.

[12] Tad Hogg and Bernardo A. Huberman. Better Than the Best: The Power of Cooperation. In *1992 Lectures in Complex Systems*, volume V, pages 165–184. Addison-Wesley, 1993.

[13] William A. Kornfeld. The Use of Parallelism to Implement a Heuristic Search. In *Proc. of the 7th Intl. Joint Conference on Artificial Intelligence (IJCAI '81)*, volume 1, pages 575–580, 1981.

[14] Teodor G. Crainic and Michel Toulouse. Parallel Meta-heuristics. In *Handbook of Metaheuristics*, chapter 17, pages 497–541. Springer, 2010.

[15] Enrique Alba, Gabriel Luque, and Sergio Nesmachnow. Parallel Metaheuristics: Recent Advances and New Trends. *International Transactions in Operational Research*, 20(1):1–48, 2013.

[16] Dariusz Barbucha. Search Modes for the Cooperative Multi-agent System Solving the Vehicle Routing Problem. *Neurocomputing*, 88:13–23, July 2012.

[17] Murray Campbell, A. Joseph Hoane Jr., and Feng-hsiung Hsu. Deep Blue. *Artificial Intelligence*, 134 (1-2):57–83, January 2002.

[18] Youssef Hamadi, Saïd Jabbour, and Lakhdar Sais. ManySAT: A Parallel SAT Solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 6(4):245–262, 2009.

[19] Diane J. Cook and R. Craig Varnell. Adaptive Parallel Iterative Deepening Search. *Journal of Artificial Intelligence Research*, 9(1):139–166, August 1998.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-151

Run-time Plan Repair for AUV Missions

Catherine HARRIS [a] and Richard DEARDEN [a] [1]
[a] *University of Birmingham, UK*

Abstract. Autonomous underwater vehicle (AUV) missions are challenging due to limited or no communication with the vehicle and uncertainty about what may be encountered during the mission. The vehicle has limited battery power and memory to store datasets and does not know how much of these resources its actions will consume. Managing the uncertainty by building contingency plans (as in previous work) is infeasible owing to the large number of contingencies. Purely online planning is also difficult because of restricted computation. We propose a mixture of off-line planning and on-line plan repair, presenting an approach for deleting parts of a plan when resources are tight, or stitching new plan fragments into an existing plan when additional resources are available. We discuss this novel approach using a simulated AUV mission domain.

1. Introduction

In recent years autonomous underwater vehicles (AUVs) have become increasingly popular for a wide variety of applications. As the cost of deploying a vehicle and the risk of loss or damage are often high, AUV missions typically consist of simple pre-scripted behaviours. Although designed to minimise risk to the vehicle and its scientific cargo, these behaviours are inevitably overly-conservative, reserving a significant proportion of battery as a contingency. Remedying this problem is difficult owing to our lack of knowledge about conditions at the ocean bottom. This makes it very hard to predict how much power a particular activity will use or how much memory datasets will consume. As improvements in battery technology allow much longer deployments, these pre-scripted missions are proving increasingly limiting. Finding ways to adapt the mission during execution has the potential to significantly improve the capabilities of these vehicles.

We consider AUV science missions, in which the vehicle collects a number of datasets from locations of interest. The AUV has limited battery power, which is required for all actions and cannot be recharged during a mission. The vehicle also has limited memory for storing data, although this can be reused if a dataset is transmitted mid-mission. A key challenge of the problem is that the resources used by each action are uncertain, so a good initial plan can fail if actions take more resource than expected, or waste resources if usage was lower than expected.

Similar problems have been discussed in the past (see Section 2). Contingency planning has been examined in multiple papers [7,13,8,2], but owing to the high uncertainty in the problem we prefer a replanning approach. However, since battery used for com-

[1] E-mail: C.A.Harris.1@cs.bham.ac.uk, richard.dearden@gmail.com

putation is no longer available for action, we aim to minimise online computation, using plan repair to efficiently update the existing plan rather than computing a new full plan.

The approach we take is to continually monitor the resource usage during execution to determine if and when updating the plan may be beneficial. Prior to execution, at key points in the plan we generate individual sub-plans for each goal in the problem, both those included in the current plan and those which may be added. These sub-plans provide an estimate of the resource cost and reward of each goal, whether adding or removing it from the plan, which we use as a heuristic to aid online plan modification. During execution, if resource usage deviates significantly from expected, the heuristic informs the selection of goals for addition or removal by representing the trade-off between resource cost and reward. Goal removal is achieved by removing all actions that only contribute to that single goal. Adding a goal involves combining the pre-generated sub-plan for the chosen goal with the existing plan, interleaving the two sets of actions until the causal structure of the resultant plan is valid. Should additional actions be required to achieve this, the system generates new plan fragments to join the two plans together. To minimise online computation, we use a classical deterministic representation for plan generation, treating resource usage as discrete (using the mean of the distribution) and then evaluate the resulting plan in the probabilistic model.

2. Related Work

The planning literature contains many examples of work with similar motivations to our own [5,6,17]. Representing a simplified version of the Mars rover domain as a Markov decision process (MDP) with continuous energy usage and time, Bresina et al. [2] compute the optimal value function for all contingencies within a branching plan. This represents the expected utility of each branch for all combinations of energy and time and can be used as a policy, dictating the optimal branch to take at run-time, given resource availability. In a follow-up paper, Feng et al. [9] use an approximate MDP approach, but this is restricted to very small problems as a value function over all continuous variables must be computed for each discrete state. Due to the high number of possible contingencies and the size of the state space in both the AUV and Mars rover domains, the use of a MDP solver is computationally infeasible for realistically-sized problems.

Solutions based on the construction and execution of branching plans are popular for domains with resource uncertainty [7,13,8,2]. Bresina and Washington [3] present a flexible on-board executive which monitors and reacts to changes in the expected utility of the rover's plan. An initial contingent schedule is constructed offline by mission operators. Contingent branches are added to define the actions to take should a predictable action failure occur during execution. Whilst our approach also augments an initial plan with alternative options prior to execution, the combination of continuous resource uncertainty and a large oversubscribed goal set in our AUV domain would require many plan branches to be generated and considered at each step in the plan. We instead delay the decision, of which goals to add or remove from the problem at each point in the plan, until run-time when the resource usage may be observed. We reduce the number of goal combinations to consider (and consequently plan branches to generate) by only considering those which are applicable given the observed resource availability.

Bidot et al. [1] present a scheduling framework capable of managing uncertainty and potential failures. An initial partial schedule is generated over a short-time horizon, based

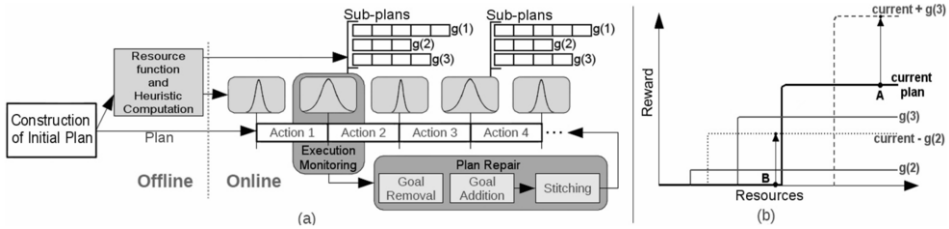


Figure 1. (a) Schematic of our approach. Execution monitoring uses the precomputed heuristic and observed resource usage to trigger plan repair. (b) Heuristic for triggering plan repair, plot shows the mean resource usage of the current plan and plan fragments: If the current resources are at A, the unused resource is sufficient to add in goal $g(3)$, increasing the expected reward. If current resources are at B (i.e. below the mean requirement of the current plan), removing $g(2)$ reduces the resource requirement and increases the expected reward.

on the most probable observations. If the duration uncertainty causes the expected value of the cost function to increase significantly, another schedule is generated online with additional ordering constraints to reduce computation. If a choice of activity is based on an observation, they delay the decision until the observation has been made, generating a new schedule for the chosen activity.

In plan repair van der Krogt and de Weerdt [19] use ideas from refinement planning [16,15], but propose an additional strategy, unrefinement planning, which allows the removal of constraints or actions which prevented goals from being met. If the current plan is no longer a solution, they create candidates for refinement by first unrefining the current plan, removing actions which depend on either the initial or goal state.

Fox et al. [10] describe plan repair as "adapting an existing plan to a new context whilst perturbing the original plan as little as possible". They define a 'plan stability' metric to measure the difference between the original and repaired plans, and adapt the local-search based planner LPG [12] to use this metric when evaluating partial plans for refinement. They compare their plan repair strategy to that of replanning and find that repair produces plans more efficiently and with much greater stability than replanning.

Nebel and Koehler [18] show that in theory, the costs of modifying an existing solution to suit a new problem are higher than generating an entirely new plan. As the AUV domain has many constraining factors—limited computational resources, a large state space— we expect online plan repair to be a more effective solution in practice, outweighing the theoretical inadequacies of plan modification.

3. Algorithm

The general structure of our approach is shown in Figure 1 (a). Offline, a plan is generated along with estimates of the expected reward as a function of resources and a heuristic to aid online plan modification. During plan execution, the actual resource usage is monitored and if there is a significant deviation, plan repair is invoked to either add or delete goals. Each step is described below. For ease of exposition, we treat a plan as a sequence of actions or interchangeably as the sequence of states that occur if the plan is executed from the initial state.

Initial Plan Generation: We assume that the initial plan is generated prior to the deployment of the vehicle when resources are plentiful. As the AUV domain is oversubscribed, the initial goals must also be selected. For each promising non-mutex combina-

tion of goals a plan is generated using an external planner. Metric-FF [14] was chosen because it is capable of generating satisficing plans for the classical deterministic representation of the AUV domain (i.e. where uncertain costs are represented using the mean of the distribution). The expected reward of each plan is then computed using Monte Carlo simulation. The plan with the highest expected reward is selected for execution. The causal links in the plan are also computed so they are available during plan repair.

Computing the Heuristic: We gain the most information about the probability of a plan completing successfully, and thus its expected reward, after performing actions with the largest resource usage uncertainty. Consequently, it is at these points that considering plan modifications is most beneficial. To aid online plan modification, we compute a heuristic prior to execution by generating a sub-plan to complete each goal in the oversubscribed problem individually. We use the expected state following the most uncertain actions as the initial state for each sub-plan, as shown in Figure 1 (a). The ratio of the estimated resource cost vs reward of adding or removing each goal at these points is used to inform the decision of which modifications to make and when.

Plan Execution Monitoring: After each action is executed the current resources are observed and used to update the expected reward of the remaining plan. If the usage was higher than expected, there may be insufficient resources to complete the plan causing the success probability and expected reward to decrease. Significant gains in expected reward can be made by removing an existing goal, thus increasing the probability of achieving the remaining goals and finishing the mission successfully. If current resources are at point B in Figure 1 (b), they are below the mean required to complete the plan (indicated by the sudden drop in expected reward) and consequently the expected reward of this plan is now very low. By removing the goal $g(2)$ from the current problem, we decrease the expected resource usage of the resulting plan (as indicated by the line labelled $current - g(2)$) whilst increasing its expected reward. If resource usage was lower than expected, the success probability (and consequently expected reward) of the remaining plan increases. This situation is illustrated by point A in Figure 1 (b), where the heuristic suggests that given the current resource levels, there are sufficient resources to accommodate the new goal $g(3)$ and still complete the existing plan. By combining the plan fragment for $g(3)$ with the existing plan (as indicated by the line labelled $current+g(3)$) the expected reward and expected resource usage of the resulting plan both increase.

Goal Removal: In removing a goal from the planning problem, we can also remove any actions which were only present to achieve that goal. Given a goal g to remove, we construct the set D_g of actions to remove by first setting $D_g = \{a_g\}$, where a_g is a dummy action with the precondition g. We then repeatedly add to D_g any action a such that all causal links from a lead to actions in D_g. All actions in D_g only contribute to the goal we wish to delete, so can be removed from the plan without impacting other goals.

Since finding D_g is fast, we select the goal to delete by computing D_g for each goal g which the heuristic suggests is plausible, starting with the goal whose sub-plan minimises the ratio of expected value to mean battery usage, to find the goal which maximises the expected value of the plan once D_g is removed. To approximate expected value, we first estimate the success probability of the remaining plan by combining the cumulative density functions (CDF) representing the resource usage of each action into a single distribution. The combined CDF represents the probability of the vehicle completing the plan given the current resources. The expected reward of a plan is the sum of the probability of achieving each goal multiplied by the reward gained. While battery and

Algorithm 1 Merging a plan fragment with an existing plan.

Input: pf—plan fragment, p—existing plan, Z—list of valid merges, initially $\emptyset$, filled during recursion.

1: **function** MERGEPLANS(Z, pf, p)
2: $valid \leftarrow false$
3: **if** p contains all goals **then**
4: $Z \leftarrow Z + p$, **return** true ▷ p is a solution
5: **else if** $pf = \emptyset$ **then return** false ▷ backtrack
6: **else**
7: $a \leftarrow$ first action in pf,
8: $pf \leftarrow pf - a$
9: $X \leftarrow$ all states in s that meet preconditions of a
10: **for all** $x \in X$ **do**
11: **if** inserting a at x causes no threats **then**
12: $merge \leftarrow p$ with a inserted at x
13: Update s, c for new action
14: $valid \leftarrow mergePlans(Z, pf, merge)$
15: **end if**
16: **end for**
17: **if** $valid$ = **false** AND a achieves no goals **then**
18: $pf \leftarrow pf - a$ ▷ skip a
19: $valid \leftarrow mergePlans(Z, pf, merge)$
20: **else**, **return** valid
21: **end if**
22: **end if**
23: **end function**

memory are not strictly independent (as battery may be used to transmit data in order to increase available memory), we assume independence for ease and speed of calculation.

Adding Goals: The system attempts to merge the plan fragment pf associated with the chosen goal and current state into the current plan p, interleaving actions to create a valid solution for the combined goal set. To create a valid merged solution, the recursive algorithm, shown in Algorithm 1, takes the first action from the new plan fragment and compares its preconditions with each state in the original plan, producing a list of candidate *merge points*—states which meet the preconditions of the action. The algorithm then checks whether causal links in the current plan are threatened by inserting the action at each merge point. If merging an action causes a threat, e.g. to the preconditions of another action, the algorithm searches the plan fragment for an action whose effect restores the threatened link and checks whether this effect is then maintained until the end of the plan fragment. If this is the case, the threat is marked as resolvable. If no links are threatened or all threats are resolvable, we add the action to the plan at the current merge point, update the plan's causal links and call the function again to insert the next action in the fragment. This continues until all valid merge points for each action in the plan fragment have been considered. Each complete merged plan is then returned for evaluation. If a threat is not resolvable, provided the action does not achieve a goal, we skip the action as it may not be required when the two plans are combined.

If no valid plan orderings are found, a 'stitching plan' is generated which uses the goal state of the sub-plan as its initial state and the unsatisfied preconditions of the remainder of the existing plan as the goal. By returning the state to this point, the stitching plan resolves all threats caused by the sub-plan. After generating the stitching plan, the two plans are again passed to Algorithm 1 to interleave them into the original plan.

4. Analysis

We designed an experiment to compare the cost of the plan modification component of our system (i.e. adding or removing a single goal, without execution monitoring triggering plan modification) to that of total replanning. For replanning, Metric-FF [14] was used in the same configuration as when generating the initial plan, sub-plans and stitch-

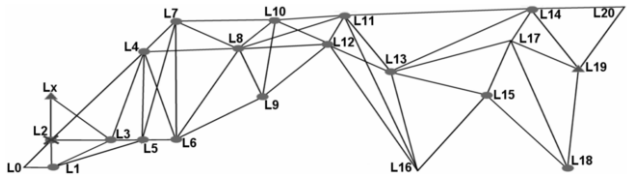


Figure 2. Connectivity of locations and datasets. Circles represent data-collection goals in the initial problem. Triangles represent goals added during runs of the experiment. The crossed $L2$ indicates the removal of this goal during the experiment.

ing plans for the plan modification algorithm, i.e. optimising for plan length. Owing to the size of the AUV domain and the inclusion of the renewable memory resource, optimising for minimal battery usage (which would produce better quality plans for this domain) was not feasible as Metric-FF does not return within a significant amount of time. We measured the time taken to construct the new PDDL [11] problem file and return either a valid solution or report a failure. We divided the experiment into two distinct tasks: firstly, to add an additional goal to the problem; and secondly, to remove an existing goal. In both cases the same initial problem (20 locations with 16 data-collection goals, shown in Figure 2) and the same initial plan (with 66 steps) were used. All updates to the goal set were made at the same point, fixed as the start of the initial plan.

As plan modifications are triggered by a change in observed resource usage, we varied the amount of battery and memory available at the point of replanning/plan-repair. When adding a goal, battery was increased from the mean usage of the existing plan, μ_b, to two standard deviations above the mean, $\mu_b + 2\sigma_b$. When removing a goal, battery was decreased from μ_b to $\mu_b - 2\sigma_b$. In both cases, memory ranges from the minimum required to complete the existing plan (501.0 units, the mean size of the largest dataset) to one standard deviation above this (501.0 $+\sigma_m$). Each data point represents the mean of 15 trials. We specified a timeout of two minutes of CPU time for each trial. A laptop with a 2.40GHz Intel Core 2 Duo CPU and 3GB of RAM was used for all runs.

Adding a goal: Both approaches were tasked with producing a valid solution when the current problem was updated to include a new data-collection goal. The first additional goal consists of collecting and delivering (either via transmission or during vehicle recovery) a dataset ($D19$ from location $L19$, see Figure 2) which is already on the route taken by the existing plan, thus requiring minimal plan modification. The second additional goal, to collect and deliver Dx from location Lx, is off the route of the existing plan, requiring much greater modification. By evaluating the performance using these two extremes, we attempted to ensure a fair representation of both approaches.

When adding the en-route data-collection goal, for $D19$, the plan modification algorithm performed well, finding a solution in 84.6% of trials, compared to replanning which found a solution in 62.1%. Replanning from scratch was, on average, 48.8 seconds slower than adding this goal using our plan modification algorithm. The modification algorithm's performance was very consistent, with a standard deviation of only 0.01 seconds, compared to 56.45 when replanning. This large difference is caused by replanning failing to return within the two-minute time-out period (as shown by the large plateau area, labelled a and b in Figure 3 (a)) and would be even greater had the timeout not been imposed. As the plan modification algorithm is not performing a full search, it is quick to report when a solution cannot be found (see areas c and b). However, as it does not consider the wider search space, plan modification may miss solutions which replanning

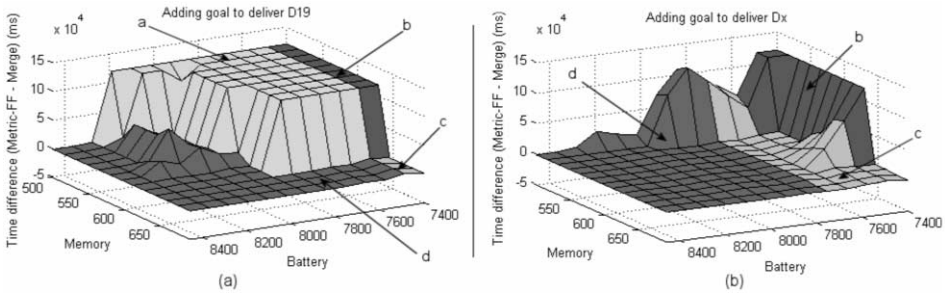


Figure 3. (a) Time difference between the two approaches as a function of available resources when adding the en-route data-collection goal $D19$. (b) Time difference when adding the off-route data-collection goal Dx. In both cases, the area labelled b represents cases where both Metric-FF and the modification algorithm failed to find a valid solution; c, cases where only plan modification failed; a, where only Metric-FF failed; and d, cases where both were successful.

was able to find (see area c). When available resources are plentiful (towards the bottom left corner of area d), replanning is very quick, out-performing the plan modification algorithm by up to 0.63 seconds. However, when both approaches found a solution, replanning was, on average, 5.98 seconds slower. We believe this difference in plan generation time is due to whether Metric-FF is able to replan using enforced-hill-climbing (as when resources are plentiful and do not significantly restrict the search) or has to resort to slower best-first search [14].

When adding the goal to deliver Dx which required the vehicle to deviate from its previous route, the performance of the plan modification algorithm was predictably worse than when adding an en-route goal, as a 'stitching' plan was required to join the plan fragment to the existing plan. Replanning returned a solution in 98.2% of cases, whereas the plan modification algorithm found a valid plan in 61.5% of cases. Plan modification was, on average, 2.83 seconds faster than replanning; however, when only considering successful runs, replanning was an average of 0.95 seconds faster. The extra time required by the plan modification algorithm in this case, compared to when adding an en-route goal is due to the generation of the stitching plan, which requires the construction of a new PDDL [11] problem file and additional actions using Metric-FF. The modification algorithm's performance was again highly consistent across all runs, with a standard deviation of only 0.02 seconds, compared to 34.38 for replanning.

Discussion: The reason stitching is required when adding the off-route goal is illustrated in Figure 5 (b), which shows a subset of the existing plan, the new plan fragment pf and the plan resulting from the merge. The only state which meets the logical preconditions of the first action in pf is state 0. However, merging this action at state 0 threatens the causal links which require the vehicle to be at $L1$ to be able to collect $D1$ and move from $L1$ to $L2$. This threat is not resolved by the remaining actions in pf and so, as $move(L0, L1)$ does not achieve any goals, we skip $move(L0, L2)$ and consider $move(L2, Lx)$. The preconditions of $move(L2, Lx)$ are met by states 3 and 4, but merging the action at either state threatens the causal link requiring the vehicle to be at $L2$ to execute $move(L2, L3)$. The algorithm skips $move(L2, Lx)$ but is then unable to add $collect(Dx)$ as its preconditions are not met by any state. Actions in pf continue to be skipped until $transmit(Dx)$ is reached. This action may not be skipped as it achieves a goal. Instead, the algorithm generates a stitching plan by calling Metric-FF using the state at the end of pf as the initial state and the unsatisfied preconditions of the exist-

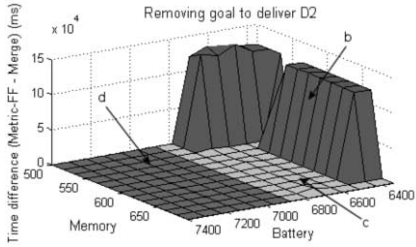


Figure 4. Time difference between the two approaches as a function of available resources when removing the goal to deliver $D2$. Area *b* represents cases where both Metric-FF and the modification algorithm failed to find a valid solution; *c*, cases where only plan modification failed; and *d*, cases where both were successful.

ing plan as the goal state. Actions added to the resulting plan from the stitching plan are shaded in Figure 5 (b). It may seem obvious to the reader that a more efficient plan would result if the vehicle was able to travel directly from $L2$ to $L1$, without having to travel via $L0$, as these are neighbours (see Figure 2). This is indeed a short-coming of the current stitching approach; however, correctly updating the variable bindings of existing actions to resolve such inefficiencies would be non-trivial as it is challenging to identify the optimal action to update without resorting to domain-specific criteria. When the full system operates as a whole, it is unlikely that the goal to return Dx would be added at this point during plan execution. The ratio of expected resource cost to expected reward is likely to be better later in the plan when the vehicle is already at $L2$, as in states 3 and 4. Consequently the decision to include this goal would be delayed until then.

When adding the en-route goal to deliver $D19$, a stitching plan is not required. Instead, the plan modification algorithm efficiently adapted the existing plan to meet the new goal by making minimal changes, as shown in Figure 5 (c). There are no states in the existing plan where the first six $move$ actions may be included without introducing threats. Since these actions do not achieve any goals, they are skipped by the merging algorithm, Algorithm 1. The preconditions of $collect(D19)$ are met by state 60 in the existing plan and, as no threats are introduced, $collect(D19)$ is added at this point. The $surface$ action causes threats and is skipped, but the goal-achieving $transmit(D19)$ action may be merged at states 62, 63 or 64, resulting in a valid plan.

Removing a goal: When removing the goal to deliver $D2$ to the scientists, replanning found a valid plan in 85.2% of cases compared to our plan modification algorithm which found a plan in 46.2% of cases (as shown in Figure 4). However, replanning from scratch was, on average, 18.31 seconds slower than modifying the plan to remove the goal (owing to replanning reaching the timeout, represented by the peaks in area b of Figure 4) and 0.13 seconds slower when comparing only successful runs. The plan modification algorithm was faster than replanning in all cases, when either finding a solution or reporting a failure. Again, the standard deviation of the plan modification algorithm (0.003 seconds) was considerably lower than for replanning (43.59 seconds).

Discussion: The reason why replanning finds more solutions than the plan modification algorithm when removing the goal to deliver $D2$ is illustrated in Figure 5 (a), which shows the subset of the existing plan concerning the completion of the goal and the resulting plan following goal removal. When removing the goal, the algorithm first removes the $transmit(D2)$ action as it was only present to complete this goal and is no longer required. The $collect(D2)$ action is also removed in this way. Note the redundancy in the resulting plan between the $move(L1, L2)$ and $move(L2, L3)$ actions,

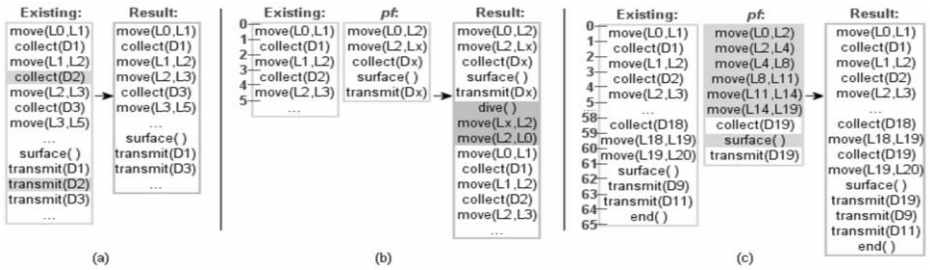


Figure 5. Plans used when: (a) removing the goal to deliver $D2$, removed actions are shaded; (b) adding the goal to deliver Dx, actions from the stitching plan are shaded; (c) adding the goal to deliver $D19$, skipped actions are shaded.

as it is possible for the vehicle to move directly from $L1$ to $L3$. As the vehicle is no longer required to collect $D2$, we theoretically have no reason to visit $L2$. However, the subsequent $move(L2, L3)$ action has a precondition that the vehicle is at $L2$ and so the $move(L1, L2)$ action is still required and may not be removed. Replanning avoids this redundancy as it is not constrained by the causal structure of the initial plan. When battery is low and memory is high, replanning may produce a plan which collects more datasets before needing to surface and transmit. This allows it to produce valid plans requiring less battery than the initial plan, causing it to succeed in a greater number of cases than plan modification. However, when operating under realistic conditions, it is unlikely that this situation would occur to the same degree. This is because the current available memory would never exceed the amount available when generating the initial plan, so replanning would be unable to capitalise by trading excess memory for savings in battery power.

5. Conclusions

We have presented a novel approach which uses a mixture of offline planning and online plan repair to produce solutions for oversubscribed domains with significant resource uncertainty. We analysed the plan modification algorithm, showing its greater speed and consistency in comparison to replanning. Minimising the time and battery required for onboard computation is important because resources used for planning are then unavailable for completing goals. As the modification algorithm is constrained by the causal structure of the initial plan, preventing large changes to the overall mission, it was unable to find solutions for some resource and goal combinations where replanning was successful. However, Fox et al. [10] argue that plan repairs should make minimal changes to the existing plan, which we consider especially true for high-risk domains such as AUV and Mars rover missions. Investigating why the AUV community has yet to widely adopt adaptive mission planning, Brito et al. [4] found uncertain vehicle behaviours to be the largest concern (39.7%) of expert AUV operators, which our approach addresses.

Our analysis of the plan modification algorithm shows promise for use within the AUV domain. Further analysis of the overall system based on complete mission runs is needed to establish the effectiveness of the full approach. As the system is domain independent, we also plan to investigate the performance of both the plan modification algorithm and the system as a whole on other oversubscribed domains, such as logistics problems.

References

[1] Julien Bidot, Thierry Vidal, Philippe Laborie, and J. Christopher Beck, 'A theoretic and practical framework for scheduling in a stochastic environment', *Journal of Scheduling*, **12**(3), 315–344, (2009).

[2] John Bresina, Richard Dearden, Nicolas Meuleau, Sailesh Ramakrishnan, David Smith, and Rich Washington, 'Planning under continuous time and resource uncertainty: A challenge for AI', in *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pp. 77–84, Alberta, Canada, (2002). Morgan Kaufmann.

[3] John L Bresina and Richard Washington, 'Robustness via run-time adaptation of contingent plans', in *Proceedings of the AAAI-2001 Spring Syposium: Robust Autonomy*, Stanford, CA, (2001).

[4] M.P. Brito, N. Bose, R. Lewis, P. Alexander, G. Griffiths, and J. Ferguson, 'The role of adaptive mission planning and control in persistent autonomous underwater vehicles presence', in *Proceedings of IEEE/OES Autonomous Underwater Vehicles (AUV)*, Southampton, (September 2012). IEEE.

[5] Rebecca Castano, Tara Estlin, Robert C. Anderson, Daniel M. Gaines, Andres Castano, Benjamin Bornstein, Caroline Chouinard, and Michele Judd, 'Oasis: Onboard autonomous science investigation system for opportunistic rover science', *Journal Field Robotics*, **24**, 379–397, (May 2007).

[6] Steve A. Chien, Russell Knight, Andre Stechert, Rob Sherwood, and Gregg Rabideau, 'Using iterative repair to improve the responsiveness of planning and scheduling', in *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling Systems*, pp. 300–307, Breckenridge, Colorado, (April 2000).

[7] A. J. Coles, 'Opportunistic branched plans to maximise utility in the presence of resource uncertainty.', in *Proceedings of the Twentieth European Conference on Artificial Intelligence*, Montpellier, France, (August 2012). IOS Press.

[8] Patrick R. Conrad, Julie A. Shah, and Brian C. Williams, 'Flexible execution of plans with choice', in *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling*, Thessaloniki, Greece, (September 2009). AAAI.

[9] Zhengzhu Feng, Richard Dearden, Nicolas Meuleau, and Richard Washington, 'Dynamic programming for structured continuous Markov decision problems', in *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pp. 154–161, Banff, Canada, (2004). AUAI Press.

[10] Maria Fox, Alfonso Gerevini, Derek Long, and Ivan Serina, 'Plan stability: Replanning versus plan repair', in *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling*, pp. 212–221. AAAI, (June 2006).

[11] Maria Fox and Derek Long, 'PDDL2.1: An extension to PDDL for expressing temporal planning domains', *Journal of AI Research*, **20**(1), 61–124, (December 2003).

[12] Alfonso Gerevini, Alessandro Saetti, and Ivan Serina, 'Planning through stochastic local search and temporal action graphs in LPG', *Journal of AI Research*, **20**, 239–290, (2003).

[13] Jonathan Gough, Maria Fox, and Derek Long, 'Plan execution under resource consumption uncertainty', in *Proceedings of the Workshop on Connecting Planning Theory with Practice at ICAPS'04*, Whistler, Canada, (June 2004). AAAI.

[14] Jörg Hoffmann, 'The Metric-FF planning system: Translating "Ignoring delete lists" to numeric state variables', *Journal of AI Research*, **20**, 291–341, (2003).

[15] Subbarao Kambhampati, 'Refinement planning as a unifying framework for plan synthesis', *AI Magazine*, **18**(2), 67–97, (1997).

[16] Subbarao Kambhampati, Craig A. Knoblock, and Qiang Yang, 'Planning as refinement search: a unified framework for evaluating design tradeoffs in partial-order planning', *Artificial Intelligence*, **76**(1-2), 167–238, (1995).

[17] D. Long, M. Woods, A. Shaw, D. Pullan, D. Barnes, and D. Price, 'On-board plan modification for opportunistic science', in *Proceedings of the IJCAI-09 Workshop on Artificial Intelligence in Space*, Pasadena, California, (July 2009). AAAI.

[18] Bernhard Nebel and Jana Koehler, 'Plan reuse versus plan generation: a theoretical and empirical analysis', *Artificial Intelligence*, **76**(1-2), 427–454, (1995).

[19] Roman van der Krogt and Mathijs de Weerdt, 'Plan repair as an extension of planning', in *Proceedings of the 15th International Conference on Automated Planning and Scheduling*, pp. 161–170, Monterey, California, (June 2005). AAAI.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-161

Embedding a Card Game Language into a General Game Playing Language

Jakub KOWALSKI [a],
[a] *Institute of Computer Science, University of Wrocław, Poland*

Abstract. We make a link between a specialized context free language expressing the rules of variety of card games, called CGDL, and the most known general-purpose game description language GDL-II. We present a systematic translation from CGDL to GDL-II, prove that the translation is correct, and analyze the complexity of resulting code in both theoretical and empirical way.

Keywords. General Game Playing, knowledge representation, Game Description Language, Card Game Description Language

1. Introduction

Artificial intelligence was always using games as test problems and demonstration of its advancement. Defeating human supremacy in popular games such as chess (*Deep Blue*), checkers (*Chinook*) or Jeopardy (*Watson*) were great achievements of AI. However, to meet the basic purpose of AI – solving general problems, a different approach should be taken. Instead of improving specialized programs which are able to play only one game, the general approach to game playing consists of developing programs which can play every game from some wide game class defined by a formal description language.

The most popular approaches in this field are related to *General Game Playing* (GGP) competition [5] and a special first order logic language based on KIF called *Game Description Language* (GDL) [10]. It has enough power do describe all turn-based, finite and deterministic n-player games with full information. Playing a game given by such a description requires not only developing a move choosing algorithm, but also implementing a reasoning approach to understand the game rules in the sense of computing legal moves, computing the state update function, and computing the goal function. Many approaches were created in this field including implementations of game playing mechanisms [2,6,7] and improving effectiveness of reasoning engines [1,8,9]. Recently, an extension of GDL called GDL-II (from *GDL with Incomplete-Information*) was proposed [15]. This language removes some restrictions of GDL and allows to describe also nondeterministic games with hidden information, but requires developing new playing techniques [13].

A card game description language developed recently in [3,4], which we call for brevity CGDL, has a unique feature – it allows one to use genetic programming to evolve game rules and so, to create new games. CGDL is a high level language with a lot of domain specific commands, to describe n-player, standard deck card games with possibility of bets (like in poker). In this paper we define a direct translation from CGDL lan-

guage into GDL-II. Then, we compare both languages as representatives of two different approaches to game description: very general one, but with difficult semantic influencing efficiency of playing algorithms, and narrow one with compact and flexible description. As GDL-II was compared with other game classes to ensure it has enough expressive power [11,12,16], no translation between other game description languages focused on practical gaming aspects is presented. We implement our translation and compare the complexity of CGDL and GDL-II both theoretically and empirically. We also distinguish features, which can possibly be transferred from CGDL into GDL/GDL-II to improve this, currently the most widely used, general game description language.

2. Preliminaries

2.1. GDL-II

GDL-II [15] is, as its predecessor GDL, strictly declarative language using logic programming-like syntax very similar to Prolog. It can describe any finite, synchronous, turn-based, n-player game. Every game description contains the declaration of players roles, the initial game state, the legal moves, the state transition function with players' percepts, the terminating conditions, and the declaration of goal function.

Language does not provide any predefined functions including arithmetic expressions or game-domain specific structures such as a board or a card deck. Every function and declaration must be defined explicitly from scratch, and the only keywords used to define game are (symbols beginning with `?` are variables):

`(role ?r)`	`?r` is a player
`random`	random player (Nature, casino, ...)
`(init ?f)`	fact `?f` is true in the initial state
`(true ?f)`	fact `?f` is true in the current state
`(legal ?r ?a)`	in the current state `?r` can perform action `?a`
`(does ?r ?a)`	`?r` performed action `?a` in the previous state
`(next ?f)`	`?f` will be true in the next state
`(sees ?r ?p)`	player `?r` will perceive `?p` in the next state
`terminal`	current state is terminal
`(goal ?r ?n)`	player `?r` score is `?n`

To be considered as *valid*, a GDL-II game specification must be *stratified* and *allowed*. This, and other syntactic restrictions ensures that the game have a unique standard model with only a finite number of true positive instances, so all deductions in Definition 1 are finite and decidable. For details we must refer to [10].

Let G be a valid GDL-II game description. It contains a finite number of function symbols and constants, which determines the set of possible ground terms Σ. Although Σ can be infinite, syntactic restrictions ensure that all sets needed to compute game flow (roles, legal moves, reachable states, etc.) are finite subsets of Σ [10]. Let $S = \{f_1, \ldots, f_k\}$ be a state of the game, denoted as the set of predicates which are true in the current position. Then we define a *base state* as $S^{\texttt{true}} \stackrel{\text{def}}{=} \{\texttt{(true } f_1\texttt{)}, \ldots, \texttt{(true } f_k\texttt{)}\}$. Let us also denote the joint move $A^{\texttt{does}} \stackrel{\text{def}}{=} \{\texttt{(does } r_1\ a_1\texttt{)}, \ldots, \texttt{(does } r_n\ a_n\texttt{)}\}$ if players $r_1, \ldots, r_n$ took actions $a_1, \ldots, a_n$. We can now introduce

Definition 1 *[15] The semantics of a valid GDL-II n player game specification G with a set of ground terms Σ is given by a state transition system composed as follows.*

- $R = \{r \in \Sigma : G \models \texttt{(role } r\texttt{)}\}$ *(player names);*
- $s_0 = \{f \in \Sigma : G \models \texttt{(init } f\texttt{)}\}$ *(initial state);*
- $t = \{S \in 2^\Sigma : G \cup S^{true} \models \texttt{terminal}\}$ *(terminal states);*
- $l = \{(r, a, S) : G \cup S^{true} \models \texttt{(legal } r\ a\texttt{)}\}$*, for all* $r \in R$*,* $a \in \Sigma$ *and* $S \in 2^\Sigma$*;*
- $u(A, S) = \{f : G \cup A^{does} \cup S^{true} \models \texttt{(next } f\texttt{)}\}$*, for all joint moves* $A : (R \mapsto \Sigma)$ *and states* $S \in 2^\Sigma$ *(state update);*
- $\mathcal{I} = \{(r, A, S, p) : G \cup A^{does} \cup S^{true} \models \texttt{(sees } r\ p\texttt{)}\}$*, for all* $r \in R \setminus \{\texttt{random}\}$*,* $A : (R \mapsto \Sigma)$*,* $S \in 2^\Sigma$ *and* $p \in \Sigma$ *(players' percepts);*
- $g = \{(r, n, S) : G \cup S^{true} \models \texttt{(goal } r\ n\texttt{)}\}$*,* $r \in R$*,* $n \in \{0, \ldots, 100\}$*,* $S \in 2^\Sigma$*.*

The execution model works as follows. Starting from the initial state s_0, in every state S every player $r \in R$ selects one legal action a such that $(r, a, S) \in l$. The `random` player choose his moves randomly with uniform probability. The joint move is applied to the state update function $u(A, S)$ to obtain a new state S'. In S', every role $r \in R \setminus \{\texttt{random}\}$ perceives every p that satisfies $(r, A, S, p) \in \mathcal{I}$. If the current state is terminal, i.e. $S \in t$, then every player obtains a score by relation $(r, n, S) \in g$ and the game ends.

The partial GDL-II game description is set as an example in Listing 2. For a more detailed language specification and full games examples we refer to [15,16].

2.2. CGDL

CGDL introduced in [4] is a context free language designed to define a rich subset of possible card games and allow to perform genetic operations to create new or evolve existing games [3]. The language domain is narrowed to n player, standard deck card games with a possibility of coin bets. All language constructions are strictly domain-dependent and use concepts of a card, number, suit, token, etc. These concepts, same as arithmetic and boolean operators, are defined a priori and used without explicit declaration.

A valid CGDL game is defined as follows. P is the number of players. Every player i has his private hand location (named Hi) and two areas for placing coins: private $Ki0$ with player's coins, and public $Ki1$ for placing bets. T is the number of virtual table locations, where cards can be placed face up. The set of game rules is organized into sequentially ordered stages, containing rules. Every stage is played in a round-robin order until all players are *out* of the game or decide to end the current stage (status set to *done*).

Every rule has a form of $modifier$ `if` $condition$ `then` $action$. The current player can perform any *action* from the set of current stage rules only if the rule *condition* is satisfied and the *modifier* limitations are met. Possible rule *modifiers* are: *computer* – must be played by the computer at the beginning of a stage, *mandatory* – must be played by a player at the beginning of a stage (after computer rules applied), *once* – can be applied only once, *optional* – no restrictions. A partial list of possible *conditions* contains

- λ – no condition to satisfy, always true;
- `sum`$,LA,R,LB$ – sums values of cards in both locations LA and LB, evaluates to true if the restriction $R \in \{<, >, =, \leq, \geq, \lambda\}$ is satisfied;
- `tokens`$,KA,R,KB$ – compares number of tokens in KA and KB using R;
- `have`$,C$ – check if the player's hand contains given card combination C, e.g. the king of hearts, three diamonds and one heart, etc.

Set of available *actions* is a superset of

- `pifr`, LA, A, F – draw a given amount A of cards from a location LA to player's hand, cards are openly visible if face F is *up*;
- `bet`, R, KA – bet an amount of tokens which, when compared to the number of tokens in the location KA, satisfies the restriction R;
- `done` / `out` – the player status is set to *done*/*out*;
- `win` – the player instantly wins the game;
- `give`, P, A – *computer* only, give A tokens to the set of players P;
- `deal`, P, A – *computer*, deal A cards from the deck to players P.

Additional special abbreviations are: HA – hands of all players, KA bets of all players, HX the hand of the current player, KX bets of the current player, `<allplayers>` rule is multiplied for all players. Rules can also define a value mapping for every card and some combination of cards (called plays). Game is over when all players are *out*, some player performs `win` action or the last stage is over. Player's score is calculated by points for every possessed token (`t`), card (`c`) and "not `out`" bonus (`s`). If player ends by choosing `win` action he got 100 and other 0. As an example, rules in Listing 1 describe a codification of game *Blackjack*. For more detailed language specification we must refer to [3,4].

3. Translation

In this section we present details of our translation. We constructed the function $\mathcal{F}$ which takes a CGDL game description $\mathcal{G}$ and returns the same game encoded in GDL-II. As the CGDL language is created by a context-free grammar, the construction is grammar based. For every game description subtree, we compute all GDL rules necessary to encode that subtree. Afterwards we remove duplicate definitions of the predicates.

The specification of the translation is provided in as much detail as we can in such limited space. As an example we translate CGDL codification of the game *Blackjack* from Listing 1 to equivalent GDL-II rules partially presented in Listing 2. References to line numbers (if not stated otherwise) refers to Listing 2.

Listing 1: CGDL codification of the game *Blackjack*.

```
1  [SETTINGS] P=3, T=0
2  [STAGES]
3    Stage 0
4  COMPUTER deal, <allplayers>, 2
5  COMPUTER give, <allplayers>, 99
6    Stage 1
7  MANDATORY if λ then bet, λ , λ
8    Stage 2
9  OPTIONAL if λ then pifr, D, 1, up
10 OPTIONAL if λ then done
11   Stage 3
12 MANDATORY if sum, HX, >, 21  then out
13 MANDATORY if sum, HX, <=, 21  then done
14   Stage 4
15 MANDATORY if sum, HX, >, HA  then gain, KA
16 [RANKING] 2:2, ..., King:10, Ace:11, Ace:1
17 [POINTS] t=1, c=0, s=0
```

Listing 2: Partial GDL-II code of the translated CGDL game *Blackjack* (in infix notation).

```
 1(role random) (role player1) ... (role player3)
...
26(init (Stage ShuffleDeck COMPUTER))
27(init (Shuffled Top))
28((init (UnShuffled ?c)) ⇐ (card ?c ?num ?suit))
29(init (ActionAvailable 0 s0a0 random COMPUTER))...
...
161((next (Token ?l2 ?n3))
162  ⇐ (true (Token ?l2 ?n1)) ∧ (movecoin ?n2 ?l1 ?l2) ∧ (asum ?n1 ?n2 ?n3))
163((next (Token ?l1 ?n3))
164  ⇐ (true (Token ?l1 ?n1)) ∧ (movecoin ?n2 ?l1 ?l2) (asub ?n1 ?n2 ?n3))
165((next (ActionAvailable ?stage ?id1 ?p ?type))
166  ⇐ (true (ActionAvailable ?stage ?id1 ?p ?type))
167  ∧ (does ?player (action ?id2 ?vis ?cond ?act)) ∧ (distinct ?id1 ?id2))
...
314((legal ?player ?act) ⇐ (tmplegal ?player ?act))
315((tmplegal ?player (action s4a0 visible (sum HX gt HA) (gain KA)))
316  ⇐ (true (Stage 4 MANDATORY)) ∧ (true (CurrentPlayer ?player))
317  ∧ (true (ActionAvailable 4 s4a0 ?player MANDATORY))
318  ∧ (not (true (PlayerStatus ?player aDONE)))
319  ∧ (not (true (PlayerStatus ?player aOUT)))
320  ∧ (handlocation ?player ?hand) ∧ (rsum ?hand ?n) ∧ (handlocation ?p1 ?h1)
321  ∧ (rsum ?h1 ?n1) ∧ (handlocation ?p2 ?h2) ∧ (rsum ?h2 ?n2)
322  ∧ (distinct ?p1 ?player) ∧ (distinct ?p2 ?player) ∧ (distinct ?p2 ?p1)
323  ∧ (or (bgt ?n ?n1) (true (PlayerStatus ?p1 aOUT)))
324  ∧ (or (bgt ?n ?n2) (true (PlayerStatus ?p2 aOUT))))
...
432((movecoin ?n ?bloc1 ?hloc) ⇐ (betlocation ?player ?hloc ?bloc)
433  ∧ (does ?player (action ?id ?vi ?cond (gain KA)))
434  ∧ (betlocation ?p ?hloc1 ?bloc1) (rKA ?n))
...
471((sees ?player (deltacoins ?n ?loc1 ?loc2)) ⇐ (movecoin ?n ?loc1 ?loc2))
472  ∧ (does ?p (action ?id visible ?cond ?action))
...
487(terminal ⇐ endstage ∧ (true (Stage ?n ?t))
488  ∧ (_stagesorder (Stage ?n ?t) ∧ (Stage EndGame none)))
489((goal ?player 100) ⇐ (true (Won ?player)))
490((goal ?player 0) ⇐ (true (Won ?p))∧(role ?player)∧(not(true (Won ?player))))
...
557((rKA ?s3) ⇐ (true (Token K11 ?n1)) ∧ (true (Token K21 ?n2))
558  ∧ (true (Token K31 ?n3)) ∧(asum ?n1 ?n2 ?s2) ∧ (asum ?s2 ?n3 ?s3))
559((rsum ?loc ?s12) ⇐ (location ?loc) ∧ (numberofcards ?loc 2)
560  ∧ (hold2cards ?loc ?c1 ?c2) ∧ (card ?c1 ?num1 ?suit1)
561  ∧ (value ?num1 ?val1) ∧ (card ?c2 ?num2 ?suit2)
562  ∧ (value ?num2 ?val2) ∧ (asum 0 ?val1 ?s1) ∧ (asum ?s1 ?val2 ?s12))
...
694(location D) ... (location H3)
695(handlocation player1 H1) ... (handlocation player3 H3)
696(betlocation player1 K10 K11) ... (betlocation player3 K30 K31)
697(card 2OfHearts 2 Hearts) ... (card AceOfSpades Ace Spades)
698(value 2 2) ... (value Ace 11) (value Ace 1)
699(stagesorder (Stage ShuffleDeck COMPUTER) (Stage 0 COMPUTER)) ...
700  (stagesorder (Stage 4 MANDATORY) (Stage EndGame none))
...
819(aplus1 0 1) ... (aplus1 100 101)
820((asum ?n 0 ?n) ⇐ (aplus1 ?n ?m))
821((asum ?n1 ?n3 ?n5) ⇐ (aplus1 ?n2 ?n3)
822  ∧ (aplus1 ?n4 ?n5) ∧ (asum ?n1 ?n2 ?n4))
823((bleq ?n ?n) ⇐ (aplus1 ?n ?m))
824((bleq ?n1 ?n3) ⇐ (aplus1 ?n1 ?n2) ∧ (bleq ?n2 ?n3))
```

Constants define relations which hold throughout the entire game (line 1 and 694). Predicate `role ?role` declares set of players `player1`, ...`player`P, `random`; `location ?loc` defines set of possible card locations: `D` for the deck, `H1`, ..., `H`P for players' hands and `T0`, ..., `T`(T–1) for table locations; `handlocation ?player ?hand` maps `player`i to his hand location `H`i. Predicate `tokenlocation ?loc` defines all token locations `K10`, ..., `K`P`0`, `K11`, ..., `K`P`1` and `betlocation ?player ?privloc ?betloc` maps these locations from player `player`i to private token location `K`i`0` and bet location `K`i`1`. List of all cards is stored in `card ?card ?number ?suit`. The card value mapping is stored in predicate `value ?card ?n`. Round-robin ordering of P players is simply defined as `playersorder ?prev ?next` relation. Predicate `stagesorder ?prev ?next` is constructed based on ordering detected in CGDL game codification.

Definition 2 *The game state is the minimal set of data necessary to distinguish that two game positions are different. CGDL game state consists of: the contents of the deck and all table and token locations; the number of current stage and player; the status of every player and the information about available computer/mandatory/once actions.*

A game state is covered by a constant number of GDL-II predicates forming $S^{\texttt{true}}$ sets and storing information necessary to encode the CGDL state according to Definition 2.

- `Stage ?id ?type` – identifier of the current stage and a type of the sub-stage containing information about the allowed actions type (*computer*, *mandatory*, *optional*); one such predicate is true at the time;
- `ActionAvailable ?stagenumber ?actionID ?player ?type` – stores all actions which can be performed by the players, if a non-optional action was made it is removed from this set and cannot be used again;
- `Token ?location ?amount` – for every `tokenlocation` stores the number of tokens in this location;
- `Table ?location ?card` – for every `tablelocation` except the deck stores the cards in this location;
- `Deck ?nextcard ?prevcard` – contains cards in the deck arranged in an order (special constant `Top` marks top of the deck);
- `CurrentPlayer ?player` – identifies the current player if it is not the dealer's turn, maximum one such predicate is true at the time;
- `PlayerStatus ?player ?status` – for every player remembers his status if necessary; status can be `aDONE` if the player decided to end the current stage, `aOUT` if he is out of the game or `mDONE` if he has no mandatory moves but is not *done*;
- `Won ?player` – true if some player performed the `win` action;
- `UnShuffled ?card` – a special predicate used at the beginning of the game to remember cards which are not yet shuffled into the deck;
- `Shuffled ?card` – a special game-beginning predicate to remember the last card chosen by the dealer to be placed at the bottom of the deck.

Definition 3 *A state is called **technical** in one of the following cases: the predicate `Stage ShuffleDeck COMPUTER` is true – which means that the virtual dealer prepares random ordering of cards in the deck; or the `CurrentPlayer` status is "out", "done" or he has no actions with fulfilled conditions but it is his turn in round-robin ordering – then every player makes the `NOOP` move and the `CurrentPlayer` shifts to the next player in order (which may lead to a next technical state).*

The initial $\mathcal{F}(\mathcal{G})$ game state s_0 (line 26) is technical, because of necessity to randomize the card deck. Every game begins with `(Stage ShuffleDeck COMPUTER)` $\in S^{\texttt{true}}$ and all cards identifiers marked as `UnShuffled`. For the next 52 turns the `random` player choose every still `UnShuffled` card with the uniform probability and put on the bottom of the actual deck. After this, `Stage` changes to the first stage from the original $\mathcal{G}$ game. Also possible players' actions are put into the `AvailableAction` predicate.

Managing legal actions can be divided into several cases. When `CurrentPlayer` is set and there is some non-`COMPUTER Stage`, the player can choose legal action from existing `tmplegal ?player (action ?id ?visibility ?condition ?action)` relations. Such relation corresponds to exactly one rule from $\mathcal{G}$. Variables `?condition` and `?action` matches CGDL rule condition and action; `?id` is an artificial identifier matching the id from `AvailableActions` relation, and `?visibility` serves to determine $\mathcal{I}$ function.

In our example, the rule from CGDL codification line 15 is translated into the `tmplegal` rule in GDL-II code line 315. Initial queries are checking stage, action availability and player's status. Then queries matching *condition* are set. In our example these are restricting sum of values of player's cards against the values of all other players cards.

Another case occurs when no `tmplegal` relation is true, due to not fulfilled conditions or exhaustion of `AvailableActions`. Then player can make only special `NOOP` move which does not change the game state. This move is also used in the following cases: for player which as not marked as `CurrentPlayers`; for player who is marked as current but he is actually *done* or *out* and for all players when there is a `COMPUTER` type stage.

For managing changes in cards and tokens possession two special predicates were introduced: `movecard` and `movecoin`. They are filling the gap between performed actions $A^{\texttt{does}}$ and state update function u. As the main idea between both predicates is similar we present here only a `movecoin` example (line 432). If relation `movecoin ?n ?from ?to` holds, this means that `?n` coins are added to `?to` coin location and subtracted from `?from` location. Multiple such relations can hold at the same time, but then their `?n` and `?to` arguments are the same. CGDL limitations implies that moving coins from multiple locations forces them to be empty, so in such cases `?n` is always set to sum of tokens in all `?from` locations. It is safe due to natural number arithmetic (subtracting from 0 is 0).

Updating the current state given players' moves, i.e. defining u function is the most complex part of the translation which, due to the limited space, we cannot describe in details (partial `next` code is shown in line 161). Every base predicate has its own set of updating rules, mostly using additional helper predicates. Sketch of the mechanics looks as follows. End of a stage occurs when all players have adequate statuses (`aOUT`, `aDONE` or `mDONE`), there is no player with any `ActionAvailable` left for the stage, or this is last turn of `ShuffleDeck` stage. If `endstage` holds, stage changes to the next stage in `stagesorder`. Similarly `CurrentPlayer` changes when `endplayer` holds. This depends on the player's actions performed, status and availability of the current stage actions.

Content of `Table`, `Deck` and `Token` is updated based on `movecard` and `movecoin` semantics. This requires several cases to examine, especially concerning taking cards from the deck without destroying its structure. If in the last turn, the player performed non-`OPTIONAL` action with some identifier, it is removed from the `ActionAvailable` set. Updating `Shuffled` and `UnShuffled` predicates takes place during the `ShuffleDeck` stage. If `does random (?shuffle ?card)` holds, then `?card` is subtracted from `UnShuffled` set and remembered as last `Shuffled` card. The predicate `Won` becomes true when the `win` action was performed.

Simulating CGDL requires natural number arithmetic for the numbers not greater then 100 (upper bound for a goal value). If some card value or play value extends that number, we can use the optional translation parameter to increase this maximum. Every created GDL-II description contains the following arithmetic and boolean functions: `aplus1`, `asum`, `asub`, `amult`, `blt`, `bgt`, `bleq`, `bgeq`, `beq`, `bneq` (line 819).

Rules defining `tmplegal` contain queries to helper predicates encoding CGDL restrictions and conditions. For example `rKA ?n` holds if `?n` is the cumulative bet of all players (line 557) and `rsum ?loc ?n` holds if values of cards in `?loc` sum up to `?n`. The function `rsum` is particularly complex and requires other helper predicates: `numberofcards ?loc ?n` which holds if in `?loc` there exactly `?n` cards, `holdcards_arity ?loc ?n` satisfied when `?loc` contains at least `?n` cards, and the predicate family `hold`n`cards ?loc ?card1 ... ?card`n which hold if there are n different cards in `?loc`. The complexity of the solution rises from a need of reasoning about the number of satisfied predicates. Example of `rsum` rule for $n = 2$ is shown in line 559.

The predicate `sees ?player ?percept` defines $\mathcal{I}$ relation, i.e. predicates perceptible by given player. Every player has a full knowledge about his hand, his private coin location, the current stage, current player and all players' statuses. Table locations, bet locations and last performed action identifier are visible to all players. Details of the action (e.g. what cards player took from the deck) are perceived only by the player who made the action or by everyone if action visibility is set to `visible` (line 471).

Definition of the `terminal` relation depends on several rules checking if: all players are *out*, some player made `win` action or `Stage EndGame none` is reached (line 487). Computing the g function is divided into two cases. If player won using the `win` action, he got 100 and all other players (including `random`) got 0. Otherwise the player's score is computed according to the CGDL game specification using `t`, `c`, `s` values.

4. Translation Properties

We state the main properties of the translation, concerning its correctness and complexity.

Definition 4 *Let S be a state of CGDL game $\mathcal{G}$. Then a state S' of game $\mathcal{F}(\mathcal{G})$ is called **corresponding**, i.e. $S \doteq S'$ if: both states have the same content and ordering of deck and every hand location, table location and token location; the actual number and type of stage; the current player; player's statuses and the set of available actions.*

Theorem 1 *For every CGDL game $\mathcal{G}$ presented translation $\mathcal{F}$ satisfies the following.*

1. *The game $\mathcal{F}(\mathcal{G})$ meet all syntactic requirements of valid GDL-II description.*
2. *The irst non-technical state of $\mathcal{F}(\mathcal{G})$ is corresponding to the first state of $\mathcal{G}$ assuming identical deck ordering.*
3. *For every non-technical game states $S \doteq S'$ there is an isomorphism between joint legal actions from both states.*
4. *For every non-technical game states $S \doteq S'$ and joint actions $A \doteq A'$, for all sequences of joint moves $\langle A', A'_2, \ldots, A'_k \rangle$ such that $S_i = u(A'_i, \ldots u(A', S') \ldots)$ and S_k is non-technical but for all $i < k$, S_i is technical, S_k is corresponding to state S after applying actions A. Such sequence always exists.*
5. *If $S \doteq S'$, S is terminal iff. $S' \in t$, goal values match for corresponding players.*

Table 1. Results of example games transformation. Visualize dependence between complexity of CGDL description (number of players, table locations, stages and rules) and resulting GDL-II code. As measurement we took number of rules and predicates used to express the game. The number of predicates and rules for base predicates are shown separately.

Game	CGDL code				GDL-II code			
					predicates		rules	
	P	T	stages	rules	base	all	base	all
Uno	2	1	2	7	10	61	38	245
Uno	3	1	2	8	10	61	39	254
Uno	3	2	2	8	10	61	39	256
Blackjack	3	0	5	12	10	63	43	361
Blackjack	3	1	5	12	10	63	43	363
Poker	3	2	13	30	10	68	61	426
Poker	4	2	13	32	10	68	63	437

Proofs of theorems are omitted due to space reasons.
We implemented a program which applies the translation function for given CGDL game description. To measure practical complexity of resulting code we provided a series of experiments using *Poker*, *Blackjack* and *Uno* games from [4] (with slightly different changes).The result of the experiments are shown in Table 1. Sizes of translated games gives a picture of complexity of their rules, which affects speed of computing GDL game states. The data also show that a number of predicates is independent on the number of players and table locations, a number of base predicates rules depends linear on the number of players and both these values have influence on overall number of rules. We state that theoretical complexity of a translated game is described by

Theorem 2 *Let $\mathcal{G}$ be a P player CGDL game description of the length N (where length is the sum of the number of stages, rules, and card/plays value mapping entries) with T table locations. Then the number of rules in the GDL-II game $\mathcal{F}(\mathcal{G})$ is $O(P + T + N)$ and the number of predicates can be bounded by a constant.*

5. Summary

The quality of solutions for General Game Playing problems heavily depends on a language which describes a game. Although the target should be to describe as many games as possible, very general languages causes two major problems. First is that providing a good playing algorithm is much harder if the type of a game is unknown. Second, that understanding and maintaining the game description is also far more complicated. Other extreme case is a language which can describe only certain types of strictly declared games, but it is high level and uses domain-dependent constructions. In this case maintaining a game description and implementing better playing algorithms is simpler.

In this paper we studied the relation between both of these approaches. On the one hand we took the most popular and the most general first order logic GDL-II language, which can describe any finite, turn-based, n-player game. On the other hand we took recently developed Card Game Description Language which is a high level language to describe card games with bets and allows a genetic manipulation on the game structure.

We constructed a translation from CGDL language into GDL-II, described details of this translation, proved its correctness and checked its complexity both theoretically and empirically. Although the size of translated game description is linear (in the sense of game rules), a complexity of simulating the game basing on pure GDL engine without compilation [9] or calling external code [14] can be computationally too hard to be performed in reasonable time, due to complicated form of queries. Our translation can be used as a benchmark tool for improving GDL players by comparison with CGDL playing algorithms. If some successful solutions could be transferred to more general approach it would be a step to reduce the gap between GDL-based and game-specific reasoners.

An interesting feature is a possibility of performing genetic operations on CGDL game code. Although CGDL was designed specifically for this case, we think it is also possible to develop this feature to GDL. GDL is a complicated language in terms of validity and semantics, but syntax rules are simple enough to make it feasible. In fact, artificially evolved GDL games could have interesting influence on creating GGP agents. They should from now on be able to play really *any* game, even codified in strange style and without common sense, not only well behaved adaptations of human games.

References

[1] Y. Björnsson and S. Schiffel. Comparison of GDL Reasoners. In *Proceedings of the IJCAI-13 Workshop on General Game Playing (GIGA'13)*, 2013.

[2] H. Finnsson and Y. Bjornsson. Simulation-based Approach to General Game Playing. In *AAAI*. AAAI Press, 2008.

[3] J. Font, T. Mahlmann, D. Manrique, and J. Togelius. Towards the automatic generation of card games through grammar-guided genetic programming. In *International Conference on the Foundations of Digital Games (FDG 2013)*, pages 360–363, 2013.

[4] J. M. Font, T. Mahlmann, D. Manrique, and J. Togelius. A Card Game Description Language. In *Applications of Evolutionary Computation*, volume 7835 of *LNCS*, pages 254–263. Springer, 2013.

[5] M. Genesereth, N. Love, and B. Pell. General game playing: Overview of the AAAI competition. *AI Magazine*, 26:62–72, 2005.

[6] S. Haufe, D. Michulke, S. Schiffel, and M. Thielscher. Knowledge-Based General Game Playing. *KI*, 25(1):25–33, 2011.

[7] P. Kissmann and S. Edelkamp. Symbolic Classification of General Multi-Player Games. In *European Conference on Artificial Intelligence (ECAI)*, volume 178 of *FAIA*, pages 905–906. IOS Press, 2008.

[8] P. Kissmann and S. Edelkamp. Instantiating General Games Using Prolog or Dependency Graphs. In *KI 2010: Advances in Artificial Intelligence*, volume 6359 of *LNCS*, pages 255–262. Springer, 2010.

[9] J. Kowalski and M. Szykuła. Game Description Language Compiler Construction. In *AI 2013: Advances in Artificial Intelligence*, volume 8272 of *LNCS*, pages 234–245. Springer, 2013.

[10] N. Love, T. Hinrichs, D. Haley, E. Schkufza, and M. Genesereth. General Game Playing: Game Description Language Specification. Technical report, Stanford Logic Group, 2008.

[11] J. Ruan and M. Thielscher. On the Comparative Expressiveness of Epistemic Models and GDL-II. In *Proceedings of the IJCAI-11 Workshop on General Game Playing (GIGA'11)*, 2011.

[12] S. Schiffel and M. Thielscher. Representing and Reasoning About the Rules of General Games With Imperfect Information. *Journal of Artificial Intelligence Research*, 49:171–206, 2014.

[13] M. Schofield, T. Cerexhe, and M. Thielscher. HyperPlay: A Solution to General Game Playing with Imperfect Information. In *AAAI Conference on Artificial Intelligence*. AAAI Press, 2012.

[14] X. Sheng and D. Thuente. Extending the General Game Playing Framework to Other Languages. In *Proceedings of the IJCAI-11 Workshop on General Game Playing (GIGA'11)*, 2011.

[15] M. Thielscher. A General Game Description Language for Incomplete Information Games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 994–999. AAAI Press, 2010.

[16] M. Thielscher. The General Game Playing Description Language is Universal. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1107–1112. AAAI Press, 2011.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-171

Effective and Efficient Identification of Persistent-state Hidden (semi-) Markov Models

Tingting LIU [a,1], and Jan LEMEIRE [a,b]
[a] *Vrije Universiteit Brussel, ETRO Dept., Pleinlaan 2, B-1050 Brussels, Belgium*
[b] *iMinds, Dept. of Multimedia Technologies (MMT), Gaston Crommenlaan 8 (Box 102), B-9050 Ghent, Belgium*

Abstract. The predominant learning strategy for H(S)MMs is local search heuristics, of which the Baum-Welch/ expectation maximization (EM) algorithm is mostly used. It is an iterative learning procedure starting with a predefined topology and randomly-chosen initial parameters. However, state-of-the-art approaches based on arbitrarily defined state numbers and parameters can cause the risk of falling into a local optima and a low convergence speed with enormous number of iterations in learning which is computationally expensive. For models with *persistent* states, i.e. states with high self-transition probabilities, we propose a segmentation-based identification approach used as a pre-identification step to approximately estimate parameters based on segmentation and clustering techniques. The identified parameters serve as input of the Baum-Welch algorithm. Moreover, the proposed approach identifies automatically the state numbers. Experimental results conducted on both synthetic and real data show that the segmentation-based identification approach can identify H(S)MMs more accurately and faster than the current Baum-Welch algorithm.

Keywords. hidden Markov models (HMMs), hidden semi-Markov models (HSMMs), Baum-Welch, local optima, model identification

1. Introduction

Hidden Markov Models (HMMs) [1] and its extension hidden semi-Markov Models (HSMMs) [2] are one of the statistical modeling tools with great success and widely used in a vast range of application fields such as audio-visual speech processing [3], machine maintenance [4], acoustics [5], biosciences [6], handwriting and text recognition [7] and image processing [8].

Classical iterative approaches (e.g., the Baum-Welch algorithm [9,10] and the gradient descent algorithm[11]) are the most commonly used methods when one wants to estimate H(S)MM parameters. However, they require a predefined number of states, which does not necessarily match the real life cases. In spite of this limitation, classical iterative approaches are still widely used to estimate H(S)MM parameters, for lack of alternatives.

[1]Corresponding Author: Tingting Liu, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium; E-mail: tliu@etro.vub.ac.be

The careless adoption of a previously known model state numbers may give misleading results. In order to solve this problem, state-of-the-art approaches decide an optimal state number either using specific criteria (e.g., the Akaike information criterion (AIC) [12], the Bayesian Information Criterion (BIC) [13]), or by structure evolving methods (e.g., the state splitting/ merging approach [14], the genetic approach [15]). However, learning H(S)MMs iteratively using the heuristic approaches above is computationally hard and often produces local optima issues. With respect to this problem, Hsu et al. [16] introduce a spectral-based algorithm for learning HMMs, which employs only a singular value decomposition and matrix multiplications, nonetheless, makes restrictive and problematic assumptions that the transition and emission matrices are full rank and the initial state vector is positive in all coordinates.

In this paper, we address the problem of model initializations and focus on models with *persistent* states (i.e., "sticky transitions"). Fox et al. [17] propose a sticky HDP-HMM which is a non-parametric, infinite-state model that automatically learns the size of state spaces and the smoothly varying dynamics robustly. However, this approach is computationally prohibitive when data sets are very large [18]. In this paper, a segmentation-based identification approach is proposed for models with *persistent* states, based on the segmentation of the observed data. Specifically, a pre-estimation step is conducted to decide the number of states and the initial model parameters approximately. This approximate estimation is served as an effective starting point of the Baum-Welch algorithm which refines the initial parameters. Consequently, both the number of iterations needed and the chance of falling into a local optimum are reduced. The improvements in effectiveness and efficiency of the proposed approach are confirmed experimentally using both simulated and real data.

The remainder of the paper is organized as follows: in Section 2, the preliminaries about HMMs and HSMMs are briefly reviewed, followed by the classifications of hidden states. Section 3 discusses the methodology of the proposed method. Experiments conducted on both synthetic and real data are described and discussed in Section 4. Finally, conclusions are given in Section 5.

2. Preliminaries

An HMM [1] is a doubly stochastic process where the underlying process is characterized by a Markov chain and unobservable (hidden) but can be observed through another stochastic process which emits the sequence of observations. Let N denote the number of states and M the number of observation symbols. Let $\mathbf{S} = \{s_1, s_2, \ldots, s_N\}$ and $\mathbf{O} = \{v_1, v_2, \ldots, v_M\}$ denote the set of states and the set of observations, respectively. Using q_t to represent the state and o_t the observation at time t, an HMM model can be characterized as below with the notation in [1]: the state transition probability matrix is $\mathbf{A} = \{a_{ij}\}$, where

$$a_{ij} = P(q_{t+1} = s_j | q_t = s_i), 1 \leq i, j \leq N \tag{1}$$

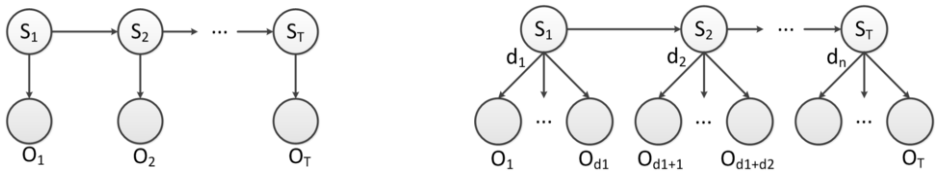
The observation probability matrix is $\mathbf{B} = \{b_j(k)\}$, where

$$b_j(k) = P(o_t = v_k | q_t = s_j), 1 \leq j \leq N, 1 \leq k \leq M \tag{2}$$

The initial state probability distribution

$$\pi_i = P(q_1 = s_i), 1 \leq i \leq N \tag{3}$$

where $s_i \in \mathbf{S}$. An HMM can be expressed with the abbreviation $\lambda = (\pi, \mathbf{A}, \mathbf{B})$. An example is shown in Figure 1a.



(a) An HMM (b) An HSMM

Figure 1. Example of an HMM and an HSMM

Hidden semi-Markov model (HSMMs) [2] is an extension of the HMMs, of which the underlying stochastic process is a semi-Markov chain instead of a Markov chain as in the HMMs. Each state has an explicit state duration variable d, which is associated the number of observations being emitted while in the state [2]. Let D denote the maximum allowed duration in a state and the state duration set as $\mathbf{D} = \{1, 2, \ldots, D\}$. For each observation sequence $o_{1:T}$, the corresponding state sequence is denoted as $s_{[1:d_1]} = i_1, s_{[d_1+1:d_1+d_2]} = i_2, \ldots, s_{[d_1+\cdots+d_{n-1}+1:d_1+\cdots+d_n]} = i_n$ and the state transitions are $(i_m, d_m) \rightarrow (i_{m+1}, d_{m+1})$, for $m = 1, \ldots, n-1$, where $\sum_{m=1}^{n} d_m = T, i_1, \ldots, i_n \in \mathbf{S}$ and $d_1, \ldots, d_n \in \mathbf{D}$. The state transition probability is defined as

$$a_{(i,d')(j,d)} = P(s_{[t+1:t+d]} = j | s_{[t-d'+1:t]} = i) \tag{4}$$

subject to $\sum_{j \in \mathbf{S} \setminus \{i\}} \sum_{d \in \mathbf{D}} a_{(i,d')(j,d)} = 1$ with zero self-transition probabilities $a_{(i,d')(j,d)} = 0$, where $i, j \in \mathbf{S}$ and $d, d' \in \mathbf{D}$. The observation probability of d observations $o_{t+1:t+d}$ being emitted in state j can be written as

$$b_{j,d}(o_{t+1:t+d}) = P(o_{[t+1:t+d]} | s_{[t+1:t+d]} = j) \tag{5}$$

The initial state probability is denoted by

$$\pi_{j,d} = P(s_{[t-d+1:t]} = j), t \leq 0, d \in \mathbf{D}. \tag{6}$$

An HSMM can be abbreviated by $\lambda = (a_{(i,d')(j,d)}, b_{j,d}(v_{k_1:k_d}), \pi_{i,d})$, where $i, j \in \mathbf{S}, d, d' \in \mathbf{D}$, and $v_{k_1:k_d}$ represents $v_{k_1}, \ldots, v_{k_d} \in \mathbf{O} \times \cdots \times \mathbf{O}$. An example is shown in Figure 1b.

A *persistent* state is a state with a high self-transition probability, i.e. the rate of remaining at the same state is high while the rates of going to other states are low. A *transient* state, on the other hand, is very likely to move to other states instead of staying at the same state. Hence the self-transition probability a_{ii} of state $i, 1 \leq i \leq N$ is used as an indicator to distinguish between persistent and transient state, i.e. if $a_{ii} > 1/N$, it is persistent, otherwise transient. This paper focuses on H(S)MMs with *persistent* states.

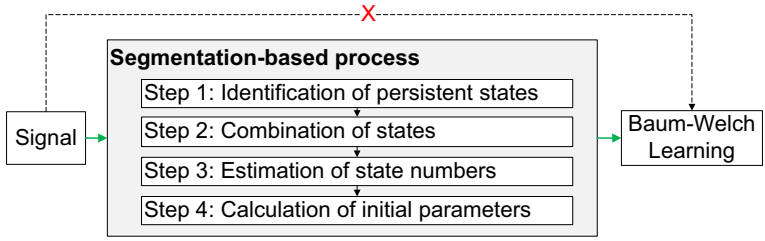


Figure 2. Scheme of the proposed approach

3. Methodology of model identification

The application we will consider is industrial machinery system maintenance which is suitable of being modeled with *persistent-state* H(S)MMs. As stated in [19], the reason of using H(S)MMs in machine maintenance and decision making is that machine operation condition can be classified into a number of meaningful states, such as "Good", "OK", "Minor defects only", "Maintenance required", "Unserviceable", so that the state definition is closer to what is used in industry and thus easy to interpret. As states determine the behavior of a system, persistence of states implies that the system will exhibit the same behavior for a certain period. Such period is called a *regime*, i.e., a time period in which the state of the system does not change, meaning the observation probabilities are constant. The assumption of state persistence is reasonable in industrial machinery systems since machine condition opt to stay in a stable and persistent state for a certain period before jumping to another state if nothing goes wrong. For instance, a machine in a "Good" condition at the current time is more likely to remain "Good" at the next time step instead of going into an "OK" condition unless the machine already degrades over a certain time period (i.e., a regime). Our algorithm is based on identifying the regimes of a state through segmentation and clustering.

The segmentation-based identification approach contains four steps: firstly, signals are split into different regimes based on different signal behaviors. Secondly, the 'similar' regimes of signal are grouped together by clustering techniques according to their similarities. The achieved labeled regimes are assumed to correspond to hidden states. Thirdly, a clustering validation index is employed to determine the number of states. Finally, H(S)MM parameters are estimated by calculating statistical occurrences of the observed signal and the estimated hidden states, then used as initial input of the standard Baum-Welch algorithm. The scheme of the methodology is shown in Fig. 2.

3.1. Step 1: Identification of persistent states by segmentation

Data sequences emitted by *persistent* states can be segmented into sub-sequences with constant behavior (observations are drawn from a stationary distribution). The transition from one state to another can be identified by detecting a difference in signal behavior. This is called a *change-point*. In this paper, we propose a sliding window-based Bayesian segmentation for splitting discrete signals by employing the test of [20]. The test calculates the Bayesian probability that two sequences have been generated by the same or by a different multinomial model. The first sequence always starts from the last change point (the first point if at the beginning) and ends at the current time point; the second

sequence is a fixed-length sliding-window starting from the next time point. If the test indicates that it is very likely (with a confidence level, for example 90%) that the two sequences are from a different model, the current time point is marked as a change point. The procedure repeats until the end of the signal. An example is shown in Figure 3.

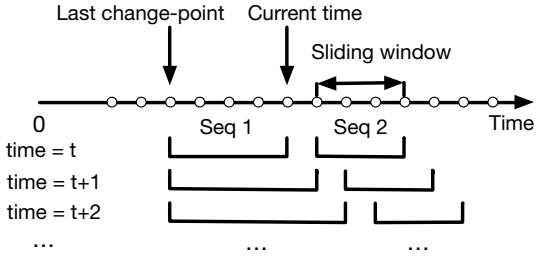


Figure 3. Sliding-window based segmentation

3.2. Step 2: Combination of states by clustering

Regimes corresponding to the same states will recur over time. Assuming there is a finite number of states, segments with the same states are detected and clustered together. In this study, the classical *k-means* clustering approach [21,22] is used to combine and label each segment, described as below: 1) feature points are obtained by averaging the data in each segment; 2) the feature points are divided into k subsequences with equal length; 3) the median values of each subsequence are used as initial starting centroids for k means clustering. Notably, 2) and 3) are the preliminary steps designed to avoid the problem of randomness in initializations of *k-means* clustering.

3.3. Step 3: Estimation of state numbers by cluster validity

In order to select the optimal number of clusters, a robust index, called Davies-Bouldin index (DBI) [23], is applied in this paper.

Suppose dataset X is partitioned into K disjoint non-empty clusters C_i and let $\{C_1, C_2, \ldots, C_K\}$ denote the obtained partitions, such that $C_i \cap C_j = \emptyset$ (empty set), $i \neq j, C_i \neq \emptyset$ and $X = \bigcup_{i=1}^{K} C_i$. The Davies-Bouldin index [23] is defined as:

$$DBI = \frac{1}{K}\sum_{i=1}^{K} \underset{i\neq j}{max}\{\frac{diam(C_i)+diam(C_j)}{dist(C_i,C_j)}\} \tag{7}$$

where $diam(C_i) = \underset{\mathbf{x}_m,\mathbf{x}_n\in C_i}{max}\{d(\mathbf{x}_m,\mathbf{x}_n)\}$ and $dist(C_i,C_j) = \underset{\mathbf{x}_m\in C_i,\mathbf{x}_n\in C_j, i\neq j}{min}\{d(\mathbf{x}_m,\mathbf{x}_n)\}$ denote the intra-cluster diameter and the inter-cluster distance, respectively. Apparently, the partition with the minimum Davies-Bouldin index is considered as the optimal choice.

3.4. Step 4: Estimation of initial parameters

The underlying assumption of our method is that segmentation of the observed signal allows us to identify quite accurately the regimes of the true model. If the regimes be-

longing to the same state are grouped correctly, these regimes offer us good insight into the behavior of the states, i.e. the observation and transition probabilities as well as the duration distribution. The probabilities are estimated based on the observed frequencies.

Parameters of an HMM (i.e., probability matrices) can be calculated by simple counting the occurrence of the observed signal and the hidden states (i.e., labels retrieved from clustering), which are computed as below [1,9,10]:

$$\bar{\pi}_i = \text{ frequency in state } s_i \text{ at time } t = 1 \tag{8}$$

$$\bar{a}_{ij} = \frac{\text{\# of trans. from } s_i \text{ to } s_j}{\text{\# of trans. from } s_i} \tag{9}$$

$$\bar{b}_j(k) = \frac{\text{\# of times in } s_j \text{ observing } v_k}{\text{\# of times in } s_j} \tag{10}$$

where *trans.* is the abbreviation for transition. Note that Baum-Welch uses the same equations in (re)-estimating model parameters. Similarly, the parameters of an HSMM can be computed as below:

$$\bar{\pi}_{i,d} = \text{ frequency in state } s_i \text{ at time } t = 1, \text{ with dur. } d \tag{11}$$

$$\bar{a}_{(i,d')(j,d)} = \frac{\text{\# of trans. from } s_i \text{ with dur. } d' \text{ to } s_j \text{ with dur. } d}{\text{\# of trans. from } s_i \text{ with dur. } d'} \tag{12}$$

$$\bar{b}_{j,d}(o_{t+1:t+d}) = \frac{\text{\# of times } o_{t+1:t+d} \text{ emitted in } s_j}{\text{\# of times in } s_j} \tag{13}$$

where *dur.* is the abbreviation for duration. The distribution of the duration d for each state can be calculated by the kernel density estimation (KDE) based on a normal kernel function [24,25]:

$$\bar{f}_h(d) = \frac{1}{\gamma h} \sum_{i=1}^{\gamma} K_N\left(\frac{d - d_i}{h}\right) \tag{14}$$

where $(d_1, d_2, \ldots, d_\gamma)$ is a duration sample drawn from a distribution with density f, K_N represents a normal kernel and h is bandwidth for the smoothing purpose, which is set as the optimal for normal densities.

4. Experimental Validation

Experiments on both synthetic and real case datasets are performed to evaluate the accuracy and efficiency of the proposed method.

4.1. Synthetic datasets

Simulated datasets are generated by 50 randomly created *persistent-state* HMMs with number of states Q (from 2 to 6), number of observations O (from 2 to 6), each combination of Q and O is repeated 2 times. Each HMM is served as a reference model and then used to generate a dataset with 4000 samples (20×200, number of observation sequences $\times$ length of sequence). The first $4/5$ observation sequences are selected as training samples and the remaining $1/5$ are used as test samples. Similar parameters are set for HSMMs with a maximum duration of $D = 30$ and datasets are generated with $T = 1000$ time steps. To identify each reference model, we train the model with both the proposed method and the standard Baum-Welch approach for comparisons. The state numbers are selected from a state pool of $[2, 2Q]$ by the Baum-Welch with the AIC criterion [12], and the proposed method with DBI cluster validation, respectively. As a result, $2Q-1$ times of the BW learning is required for each learning task. On the contrary, the proposed method starts with a pre-learned initial parameters, hence requires only 1 time. In the segmentation step of the proposed method, the window size and the confidence level can be adjusted according to different applications, which here are set to 20 and 0.9 empirically.

The comparisons are conducted on two aspects: learning accuracy and speed. The accuracy in model identification is evaluated by comparing the log-likelihoods (LL) difference with the reference model on test samples. The LL of the observation $o_{1:T}$ given the model λ measures how well the model fits the data ($\log(P(o_{1:T}|\lambda))$. If the difference between the likelihoods is below a certain threshold (5% in this paper), the model is considered as correctly learned; otherwise the learned model is assumed to be trapped in a local optimum. The learning speed is compared on the total time of learning (measured in seconds) and the number of iterations to converge.

Table 1. Performance on synthetic data for standard Baum-Welch algorithm and proposed method

Models		HMM		HSMM	
Criteria		Random BW + AIC	Proposed method	Random BW + AIC	Proposed method
Accuracy	Test-set LL difference (%)	18.7	2.6	46.27	39.37
	Test-set local optima (%)	39.4	14.0	90.20	72.00
Speed	Average learning time (seconds)	13.32	2.62	2.44	1.55
	Average number of iterations (#)	25.94	8.70	4.32	5.56

Experiment results in Table 1 show an obvious improvement of the proposed method compared to the Baum-Welch algorithm for HMMs: the model distance with the reference model and the number of local optima are lowered and with a faster learning speed and fewer number of iterations to converge. For the HSMMs, improvements can be seen marginally.

4.2. Bearing dataset

The proposed method is applied to a bearing dataset for machine maintenance provided by the POM project[2]. The set-up consists of steel cord production machines located in the production plant in China. These machines were continuously monitored for bearing degradation using accelerometers and temperature sensors. The temperature are logged regularly by the temperature sensors on both sides, i.e., 'input' and 'output' side. A temperature overshoot protection is implemented in the machine's controllers in order to avoid catastrophic failures of bearings. When temperature exceeds the temperature threshold for more than predefined observation time, the machine stops and restarts when the temperature decreases below the temperature threshold.

The temperature evolution is predicted by one of the POM project partners in a run-by-run way, on which regression is applied taking context feature into account by Dynamic Time Warping. However, each run is learned separately without considering the run-dependence. To address this issue, we use the H(S)MM models to study the dynamic nature and correct the prediction errors of the regression method by the inferred model. In the experiment, the dataset is the prediction errors of the 'input' temperature signal from one of the machines at the end of each run, which contains 3160 data points. The state number selection of the BW and the proposed methods is conducted on a state pool of [2, 8] via the AIC criterion and the DBI index, respectively. To lower the side affect of the randomness in the BW initialization, the learning is repeated 5 times for the BW and the averaged performance is used to compare with the proposed method.

Figure 4a shows the learning results of the proposed method: segmentation via change-point detection (above) and the labeled hidden states by the K-means clustering (below). The clustering validation by the DBI index is shown in Figure 4b of which the one with 8 clusters (states) with the minimum DBI value is selected as an optimal choice. Results suggest the feasibility of combining the states with 'similar' behaviors by clustering. Performances of both approaches are shown in Table 2. The averaged log-

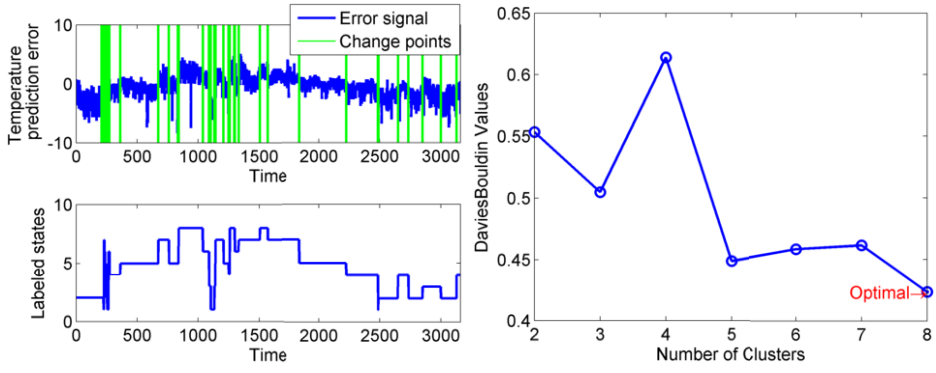


Figure 4. Pre-estimation results of the proposed method. (a) Segmentation via change-point detection and labeling via clustering. (b) The DBI indexes.

likelihood values of the Baum-Welch method are lower and the speed is dramatically

[2]www.pom-sbo.org

Table 2. Comparisons of the average learning performance over repetitions on the bearing data

Models	HMM		HSMM	
Criteria	Random BW + AIC	Proposed method	Random BW + AIC	Proposed method
Number of states	8	8	7	8
Average log-likelihood	-2.0284	-2.0175	-2.0523	-2.0149
Average learning time (seconds)	205.3406	9.0711	163.6585	7.1672

slowed down compared to the proposed method. The reasons for the speed gain are explicit: both methods select the state numbers from a state pool from 2 to 8. However, the proposed method uses a clustering validation index with only one run of Baum-Welch learning, instead of 7 runs for the traditional BW method; moreover, even the average duration of runs in traditional BW, is still much longer than proposed method, because a rather 'accurate' initialization of the proposed method requires not only fewer iterations, but also less time to converge.

5. Conclusions

This paper introduces an extension to the current algorithm for H(S)MMs identification based on segmentation and clustering techniques. Both state number selection and parameters initialization are addressed. Enhancement in the accuracy of H(S)MMs identification and the learning converge speed are achieved through the development of a pre-estimation step which avoids the local optimal problem. The effectiveness and efficiency of the proposed method are confirmed through experiments on both synthetic and real signals. Future work will improve the proposed method and extend it to more types of models (e.g., the ones with *non-persistent* states), as well as consider an application difference for a better quantification of the model parameters estimation.

References

[1] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, pp. 257–286, 1989.

[2] S. zheng Yu, "Hidden semi-markov models," *Artificial Intelligence*, 2010.

[3] A. Verma, N. Rajput, and L. Subramaniam, "Using viseme based acoustic models for speech driven lip synthesis," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, vol. 5, pp. V–720, IEEE, 2003.

[4] J. bo Yu, "Health condition monitoring of machines based on hidden markov model and contribution analysis," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, pp. 2200 – 2211, 2012.

[5] B. Logan and P. Moreno, "Factorial hmms for acoustic modeling," in *International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, 1998.

[6] R. J. Boys, D. A. Henderson, and D. J. Wilkinson, "Detecting Homogeneous Segments in DNA Sequences by Using Hidden Markov Models," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 49, no. 2, pp. 269–285, 2000.

[7] A. Fischer, K. Riesen, and H. Bunke, "Graph similarity features for hmm-based handwriting recognition in historical documents," in *2010 12th International Conference on Frontiers in Handwriting Recognition*, vol. 0, pp. 253–258, IEEE, Nov. 2010.

[8] J. Li, A. Najmi, and R. M. Gray, "Image classification by a two dimensional hidden markov model," *IEEE Transactions on Signal Processing*, vol. 48, 2000.

[9] L. E. Baum and T. Petrie, "Statistical Inference for Probabilistic Functions of Finite State Markov Chains," *The Annals of Mathematical Statistics*, vol. 37, pp. 1554–1563, 1966.

[10] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *The Annals of Mathematical Statistics*, vol. 41, pp. 164–171, 1970.

[11] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a markov proces to automatic speech recognition," 1983.

[12] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *Second International Symposium on Information Theory* (B. N. Petrov and F. Csaki, eds.), pp. 267–281, Akadémiai Kiado, 1973.

[13] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[14] M. Ostendorf and H. Singer, "HMM topology design using maximum likelihood successive state splitting," *Computer Speech and Language*, vol. 11, pp. 17–41, 1997.

[15] J. Goh, L. Tang, and L. A. turk, "Evolving the structure of Hidden Markov models for micro aneurysms detection," in *UK Workshop on Computational Intelligence*, 2010.

[16] D. Hsu, S. M. Kakade, and T. Zhang, "A Spectral Algorithm for Learning Hidden Markov Models," *Computing Research Repository*, vol. abs/0811.4, 2008.

[17] E. B. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky, "An hdp-hmm for systems with state persistence," in *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, (New York, NY, USA), pp. 312–319, ACM, 2008.

[18] L. Du, M. Chen, J. Lucas, and L. Carin, "Sticky hidden markov modeling of comparative genomic hybridization," *Signal Processing, IEEE Transactions on*, vol. 58, pp. 5353–5368, Oct 2010.

[19] X.-S. Si, W. Wang, C.-H. Hu, and D.-H. Zhou, "Remaining useful life estimation - A review on the statistical data driven approaches," *European Journal of Operational Research*, vol. 213, pp. 1–14, August 2011.

[20] M. Johansson and T. Olofsson, "Bayesian Model Selection for Markov, Hidden Markov, and Multinomial Models," *IEEE Signal Processing Letters*, vol. 14, pp. 129–132, 2007.

[21] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob.*, vol. 1, pp. 281–297, Univ. of Calif. Press, 1967.

[22] E. W. Forgy, "Cluster analysis of multivariate data: efficiency versus interpretability of classifications," *Biometrics*, vol. 21, pp. 768–769, 1965.

[23] D. L. Davies and D. W. Bouldin, "A Cluster Separation Measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[24] M. Rosenblatt, "Remarks on Some Nonparametric Estimates of a Density Function," *The Annals of Mathematical Statistics*, vol. 27, pp. 832–837, Sept. 1956.

[25] E. Parzen, "On Estimation of a Probability Density Function and Mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-181

Supervised Separation of Speech from Background Piano Music using a Nonnegative Matrix Factorization Approach

A. MARTINEZ-COLÓN [a,1], F. J. CANADAS-QUESADA [a] and P. VERA-CANDEAS [a] and N. RUIZ-REYES [a] and F. MORENO-FUENTES [a]
[a] *Telecommunication Engineering Department, University of Jaén, Spain*

Abstract. This paper presents a supervised algorithm for separating speech from background non-stationary noise (piano music) in single-channel recordings. The proposed algorithm, based on a nonnegative matrix factorization (NMF) approach, is able to extract speech sounds from isolated or chords piano sounds learning the set of spectral patterns generated by independent syllables and piano notes. Moroever, a sparsity constraint is used to improve the quality of the separated signals. Our proposal was tested using several audio mixtures composed of real-world piano recordings and Spanish speech showing promising results.

Keywords. Sound separation, Non-negative matrix factorization, training, supervised, sparse, interference

1. INTRODUCTION

Separation of a target source (speech) from background non-stationary noise (piano) is still a challenging problem in artificial intelligence, signal processing and music research. The speech refers to vocal sounds used in a human communication whereas the piano sound refers to the sounds generated by a piano instrument.

Several approaches to separate speech and background non-stationary noise have been proposed in the last years [1] [2] [3]. Schmidt et.al [1] presented a method, based on non-negative sparse coding, for reducing wind noise in recordings of speech based on a pre-estimated source model only for the noise. In [2], a sparse latent variable model is proposed which can be employed for the decomposition of time/frequency distributions to perform separation of sources from single-channel recordings. In [3], speech is modeled using a non-negative hidden Markov model, which uses multiple non-negative dictionaries and a Markov chain to jointly model spectral structure and temporal dynamics of speech.

Non-negative matrix factorization (NMF) has been successfully applied in the field of speech and music processing in recent years [4] [5] [6] [7] [8] [9]. Lee and Seung

[1]Corresponding Author: A. Martinez-Colón, Telecommunication Engineering Department, University of Jaén, Spain ; E-mail: fcanadas@ujaen.es

[10] [11] developed standard NMF, a technique for multivariate data analysis in which an input magnitude spectrogram, represented by a matrix X, is decomposed into the product of two non-negative matrices W and H,

$$X \approx WH \tag{1}$$

where each column of the basis matrix W represents a spectral pattern from an active sound source. Each row of the gains matrix H represents the time-varying activations of a spectral pattern factorized in the basis matrix. In general, NMF approaches can be classified into three categories [12]

Supervised: all spectral patterns both the target and non-target source are trained previously to the separation stage.

Semisupervised: only spectral patterns from the target source or non-target source are trained previously to the separation stage.

Unsupervised: no training stage is used. Instead, the factorization process is performed using different type of constraints.

In this work, we propose a supervised NMF approach to separate speech and polyphonic piano music in single-channel recordings. Our proposal is composed of two stages: training and separation. In the training stage, the system learn the spectral patterns from sounds related to syllables of Spanish speech and sounds from musical isolated piano notes. Using the previous patterns, our proposed algorithm is able to decompose a monaural audio mixture into speech and piano signals. As it will be explained later, we have used a sparsity constraint in order to improve the quality of the speech and minimizing the interference of the piano and vice versa.

This paper is organized as follows. In section 2, the proposed method is depicted in detail. In section 3, test data, experimental setup and metrics are explained. In section 4, experimental results are shown. Finally, the conclusions and future work are presented in section 5.

2. PROPOSED METHOD

The scheme of the proposed method is shown in Figure 1. Because of our proposed method is based on a supervised NMF approach, it needs a two training stages. The first one is related to factorize the spectral patterns of the syllables of the speech. The second one is related to factorize the spectral patterns of the piano notes. The most used cost functions are the Euclidean (EUC) distance, the generalised Kullback-Leibler (KL) and the Itakura-Saito (IS) divergences. However, in this work, the KL and IS divergences have been analyzed because they have provided the best results in the separation stage.

2.1. Speech training stage

To obtain the spectral patterns W_s of the speech, a speech database D_s was generated recording, using a portable recorder *Zoom H4n* [13], a set of different syllables of the Spanish language. Specifically, the speech database is composed of 420 syllables: 5 syllables of one letter, 118 syllables of two letters, 291 syllables of three letters and 6 sylla-

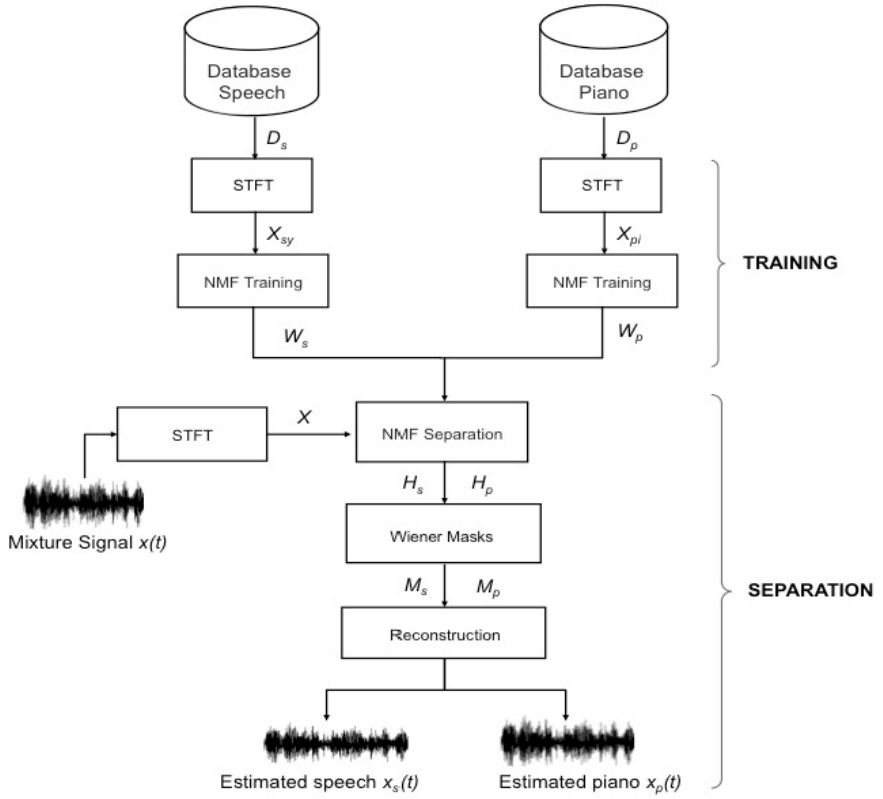


Figure 1. Overview of the proposed supervised NMF approach

bles of four letters. The selection of the syllables was made taking into account the most likely syllables to be spoken in the Spanish language. In the factorization process, we have considered K_s spectral patterns to model each syllable.

In order to estimate the speech basis W_s or gains H_s matrices, the iterative algorithm proposed in [11] [12] can be applied,

- *Kullback-Leibler* divergence

$$W_s = W_s \odot \frac{\left(X_{sy} \odot (W_s \cdot H_s)^{-1}\right) \cdot H_s^T}{\mathbf{1} \cdot H_s^T} \tag{2}$$

$$H_s = H_s \odot \frac{W_s^T \cdot (X_{sy} \odot (W_s \cdot H_s)^{-1})}{W_s^T \cdot \mathbf{1}} \tag{3}$$

- *Itakura-Saito* divergence

$$W_s = W_s \odot \frac{\left(X_{sy} \odot (W_s \cdot H_s)^{-2}\right) \cdot H_s^T}{(W_s \cdot H_s)^{-1} \cdot H_s^T} \tag{4}$$

$$H_s = H_s \odot \frac{W_s^T \cdot \left(X_{sy} \odot (W_s \cdot H_s)^{-2}\right)}{W_s^T \cdot (W_s \cdot H_s)^{-1}} \tag{5}$$

where $\odot$ is the element-wise product operator, ^{T} is the transpose operator, X_{sy} is the magnitude spectrogram of each syllable and **1** is an all-one elements matrix. The speech training procedure is summarized in Algorithm 1

Algorithm 1 Training of Speech Spectral Patterns

1 **for** each syllable **do**
2 Compute the speech magnitude spectrogram X_{sy} from a syllable of the database D_s.
3 Initialise all rows of the gain matrix H_s with random positive values.
4 Initialise all columns of the basis matrix W_s with random positive values.
5 Update bases W_s using eq. (2) or (4)
6 Update gains H_s using eq. (3) or (5)
7 Repeat steps 5-6 until the algorithm converges (or the maximum number of iterations *MaxIter* is reached).
8 **end for**

2.2. Piano training stage

To obtain each spectral patterns W_p of a piano instrument, the piano database D_p was generated using samples of notes from a piano instrument [14]. Specifically, the piano database is composed of 88 sounds of isolated piano notes played on a normal intensity. In the factorization process, we have considered K_p spectral patterns to model each piano note.

The piano update rules to compute W_p and H_p are similar to speech ones (see eq. (2-5)) replacing X_{sy} for the magnitude spectrogram X_{pi} of each musical note from a piano instrument and replacing W_s to W_p and H_s to H_p. The piano training procedure is summarized in Algorithm 2

Algorithm 2 Training of Piano Spectral Patterns

1 **for** each note **do**
2 Compute the piano magnitude spectrogram X_{pi} from a piano note of the database D_p.
3 Initialise all rows of the gain matrix H_p with random positive values.
4 Initialise all columns of the basis matrix W_p with random positive values.
5 Update bases W_p using eq. (2) or (4)
6 Update gains H_p using eq. (3) or (5)
7 Repeat steps 5-6 until the algorithm converges (or the maximum number of iterations *MaxIter* is reached).
8 **end for**

As a consequence of using a supervised NMF approach, W_s and W_p are pre-computed and known in the training stages and held fixed during the factorization process in the separation stage.

2.3. Separation stage

The magnitude spectrogram X of a mixture signal $x(t)$ can be performed by the Short-Time Fourier Transform (STFT) using a N samples *Hamming* window and J samples time shift. The mixture spectrogram X is composed of a speech X_s and a piano X_p spectrograms,

$$X = X_s + X_p \tag{6}$$

, where each spectrogram X_s or X_p represents the specific spectral features exhibited by the speech and piano instrument. In this manner, our factorization model is defined

$$\hat{X} \approx \hat{X}_s + \hat{X}_p \approx (W_s * H_s) + (W_p * H_p) \tag{7}$$

being $\hat{X}$, $\hat{X}_s$, $\hat{X}_p$, W_s, W_p, H_s and H_p the estimated mixture spectrogram, the estimated speech spectrogram, the estimated piano spectrogram, the speech and piano spectral patterns and the speech and piano gains.

The speech H_s and piano H_p gains update rules are shown using the *Kullback-Liebler* divergence (eq. (8) and (9)) and *Itakura-Saito* divergence (eq. (10) and (11)) [12] with a sparsity (speech λ_s or piano λ_p) constraint [4]

$$H_s = H_s \odot \frac{W_s^T \cdot (X \odot ((W_s \cdot H_s) + (W_p \cdot H_p))^{-1})}{W_s^T \cdot 1 + \lambda_s} \tag{8}$$

$$H_p = H_p \odot \frac{W_p^T \cdot (X \odot ((W_s \cdot H_s) + (W_p \cdot H_p))^{-1})}{W_p^T \cdot 1 + \lambda_p} \tag{9}$$

$$H_s = H_s \odot \frac{W_s^T \cdot (X \odot ((W_s \cdot H_s) + (W_p \cdot H_p))^{-2})}{W_s^T \cdot ((W_s \cdot H_s) + (W_p \cdot H_p))^{-1} + \lambda_s} \tag{10}$$

$$H_p = H_p \odot \frac{W_p^T \cdot (X \odot ((W_s \cdot H_s) + (W_p \cdot H_p))^{-2})}{W_p^T \cdot ((W_s \cdot H_s) + (W_p \cdot H_p))^{-1} + \lambda_p} \tag{11}$$

where $\odot$ is the element-wise product operator and the ^{T} is the transpose operator .

Once the update rules have been performed, the estimated spectrograms $\hat{X}_s$ and $\hat{X}_p$ are used to compute soft masking M_s (speech) and M_p (piano) (Wiener masking) since it provides less artifacts in the resynthesis but increases the amount of interference between speech and piano.

$$\mathbf{M}_s = \frac{\hat{\mathbf{X}}_s}{\hat{\mathbf{X}}_s + \hat{\mathbf{X}}_p} \tag{12}$$

$$\mathbf{M}_p = \frac{\hat{\mathbf{X}}_p}{\hat{\mathbf{X}}_s + \hat{\mathbf{X}}_p} \tag{13}$$

The phase information related to the speech is computed by multiplying the mask M_s with the complex spectrogram related to the mixture signal $x(t)$. The inverse transform is then applied to obtain an estimation of the speech signal $x_s(t)$. The computation of $x_p(t)$ is performed in a similar procedure taking into account M_p. In algorithmic approximation, the separation procedure is detailed in Algorithm 3.

Algorithm 3 Speech and Piano Separation

1 Compute the magnitude spectrogram X of the mixture signal.
2 Initialise H_s and H_p with random nonnegative values.
3 Initialise W_s and W_p from the training stage.
4 Update H_s using eq. (8) or eq. (10)
5 Update H_p using eq. (9) or eq. (11)
6 Repeat steps 4-5 until the algorithm converges (or the maximum number of iterations $MaxIter$ is reached).
7 Reconstruction of the estimated speech signal $x_s(t)$
8 Reconstruction of the estimated piano signal $x_p(t)$

3. EVALUATION

3.1. Test data

To evaluate the performance of the proposed method, we have created a test database D composed of 10 mixtures signals. Each mixture signal is composed of a 20 seconds duration speech and polyphonic piano excerpt. Each piano excerpt has been randomly extracted from the MAPS database [15]. Each speech excerpt has been randomly extracted from a set of 44 sentences spoken by a Spanish speaker. From these sentences, we have selected 10 excerpts of 20 seconds duration. Highlight that the set of syllables and piano notes used in the training are not the same used in the test in order to validate the results.

To evaluate different acoustic scenarios, the test database D has been mixed using -5, 0 and 5 dB of signal-to-noise ratio (see Table 1).

Table 1. Acoustic scenarios in the evaluation process.

Name	SNR(dB)
D_{-5}	-5
D_0	0
D_5	5

3.2. Experimental setup

The proposed method has been tested using different configurations of parameters: $N = (4096, 2048, 1024)$, $J = (2048, 1024)$, $maxIter = (100, 200, 300, 500)$. However, we have used $N = 4096(93ms)$, $J = 1024(23ms)$ and $maxIter = 100$ because a preliminary study showed that that configuration showed better results and lower computational cost.

Assuming the previous configuration (N, J and $maxIter$), separation results will be analyzed taking into account the type of divergence (KL or IS), the number of spectral patterns $K_s - K_p = (1-1, 3-1, 3-3, 5-1, 5-5, 10-1)$ and the sparsity parameters $\lambda_s - \lambda_p = (0-0, 0-1, 0.3-0.7, 0.5-0.5, 0.5-0.7, 0.5-1, 0.7-0.3, 0.7-0.5, 1-0, 1-0.5, 1-1)$.

3.3. Metrics

Three metrics [16] are used to measure the performance of the proposed method: source-to-distortion ratio (SDR), which reports about the overall quality of the separation process; source-to-interferences ratio (SIR), which provides a measure of the presence of piano sounds in the speech signal and vice versa; and source-to-artifacts ratio (SAR), which reports about the artifacts in the separated signal due to separation and/or resynthesis.

4. EXPERIMENTAL RESULTS

The proposed method was evaluated using all the possible combinations (type of divergence, number of spectral patterns and sparsity parameters) explained in section 3.2. Experimental results indicated that the best results were obtained using the optimal configurations shown in Table 2.

Table 2. Optimal configurations for speech-piano separation

Name	Divergence	K_s	K_p	λ_s	λ_p
C_1	IS	5	5	1	1
C_2	KL	5	5	1	0.5

The optimal configurations C_1 and C_2 use a number $K_s - K_p = 5$ speech and piano spectral patterns because this is the minimum number of patterns to model the spectral diversity exhibited by speech and piano (in a less proportion). Moreover, both configurations show the sparsity constraint active to improve the quality of the speech. As a consequence of the monophonic feature of speech, the speech sparsity parameter λ_s is higher that piano sparsity λ_p because speech is more sparse than piano instrument. As a example, a 6-seconds mixture spectrogram (Figure 2) and the output of the proposed method (Figure 3) using the configuration C_2 are shown. It can be observed how our proposal has successfully extracted the main features of the speech sounds in the estimated spectrogram.

Separation results are shown in Figure 4 in which the standard NMF ($\lambda_s = \lambda_p = 0$), the optimal configurations C_1, C_2 and the ideal case are compared. The ideal case shows the best SDR, SIR and SAR since in this case, the estimated speech is composed of the speech used in the mixing process to create the test database. It can be seen how all metrics (SDR, SIR and SAR) increase to evaluate a more ideal acoustic scenario. This fact is because our system performs better separation when the speech exhibits a higher power compared to the piano signal. In the three acoustic scenarios we can observe

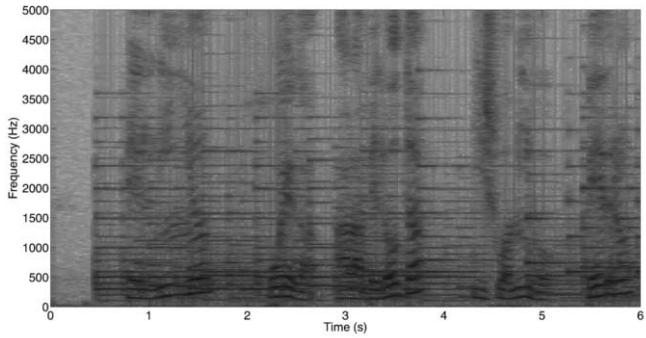


Figure 2. Time-frequency representation of a mixture composed of speech and piano sounds

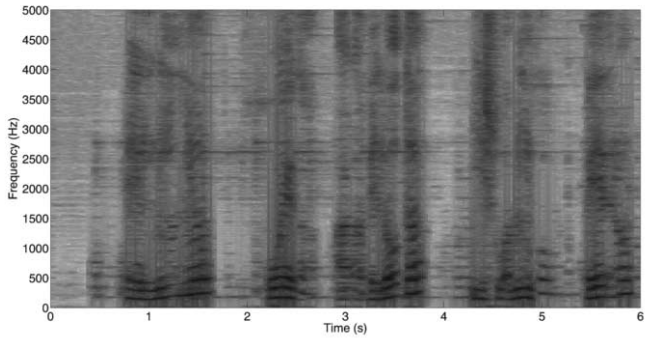


Figure 3. Time-frequency representation of the estimated speech using the configuration C_2

the configuration C_1 achieves the best SDR-SAR results considering the quality of the estimated speech but the speech contains a higher interference from piano. However, the configuration C_2 provides worse results taking into account the quality of the estimated speech (the speech is still clearly intelligible) but a lower interference from piano. Under our opinion, both configurations C_1 and C_2 can be selected as the best one because the fundamental criterion depends on the subjective quality provided by the highest SDR-SAR or SIR to each listener.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have developed a system for separating speech from background non-stacionary noise (polyphonic piano music) in single-channel recordings. Our system, based on a supervised NMF approach, is able to learn most of spectral patterns of the syllables of the Spanish speech and the spectral patterns of the piano notes. Moreover, a sparsity constraint has been modeled to improve the separation results. An advantage of our system is its flexibility to analyze another type of non-stationary noise replacing the spectral patterns of piano by the spectral patterns of the specific non-stationary noise.

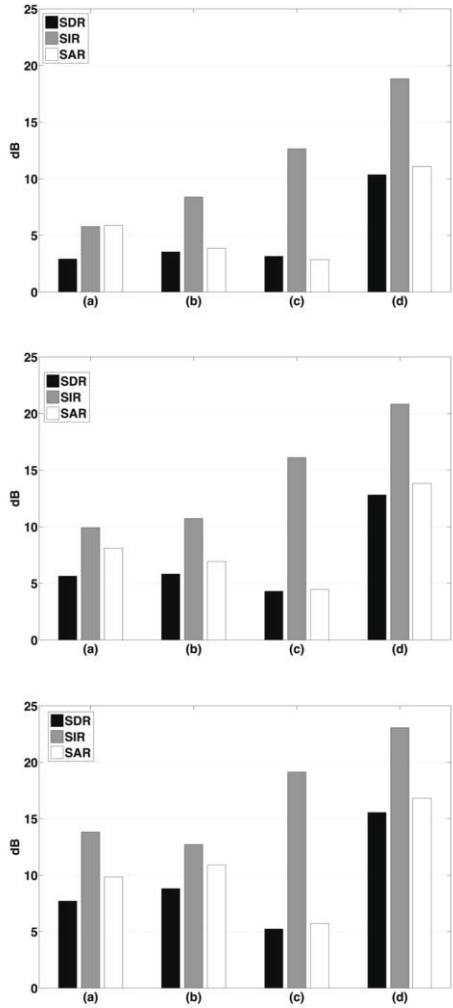


Figure 4. SDR-SIR-SAR results comparing: (a) the standard NMF ($\lambda_s = \lambda_p = 0$), (b) Configuration C_1, c) Configuration C_2 and (d) Ideal case. Test database D_{-5} (top); Test database D_0 (middle); c) Test database D_5 (bottom)

Results show that a small number of speech and piano spectral patterns is needed to model the spectral diversity exhibited by speech. The optimal configurations use the sparsity constraint to improve the quality of the speech. The configuration C_1 is the best considering the quality of the estimated speech but the configuration C_2 is the best one taking into account the minimum interference from piano.

Our future work will be focused on two topics. Firstly, developing a semi-supervised approach in order to allow the system to learn the unknown patterns active in the mixture. Secondly, a study of the influence of the speech spectral patterns in the performance of the separation taking into account different voices of different vocal characteristics.

ACKNOWLEDGEMENTS

This work was supported by the Andalusian Business, Science and Innovation Council under project P2010- TIC-6762 and (FEDER) the Spanish Ministry of Economy and Competitiveness under Project TEC2012-38142-C04-03.

References

[1] M. N. Schmidt, Jan Larsen and Fu-Tien Hsiao. Wind Noise Reduction Using Non-Negative Sparse Coding, Conference: IEEE Workshop on Machine Learning for SignalProcessing-MLSP ,2007.

[2] P. Smaragdis, B. Raj and M. Shashanka. Supervised and Semi-Supervised Separation of Sounds from Single-Channel Mixtures. Mitsubishi Electric Research Laboratories Cambridge MA, USA. Department of Cognitive and Neural Systems Boston University, Boston MA, USA. 2007.

[3] G. Mysore and P. Smaragdis. A non-negative approach to semi-supervised separation of speech from noise with the use of temporal dynamics. In Proceedings Interntational Conference on Acoustics, Speech and Signal Processing (ICASSP). Prague, Czech Republic, May, 2011

[4] M. Schmidt and R. Olsson. Single-channel speech separation using sparse non-negative matrix factorization, in *Spoken Language Proceesing, ISCA International Conference on (INTERSPEECH)*, 2006.

[5] T. Virtanen. 'Monaural Sound Source Separation by Non-Negative Matrix Factorization with Temporal Continuity and Sparseness Criteria", *IEEE Transactions on Audio, Speech, and Language Processing*, no.3, vol 15, March 2007.

[6] K. W. Wilson, B. Raj and P. Smaragdis, 2008. Regularized Non-Negative Matrix Factorization with Temporal Dependencies for Speech Denoising. In proceedings of Interspeech 2008, Brisbane, Australia, September 2008

[7] J. So-Young, K. Kyuhong, J. Jae-Hoon and C. Kwang. Semi-blind disjoint non-negative matrix factorization for extracting target source from single channel noisy mixture, *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, 2009

[8] E. Grais, H. Erdogan. Single Channel Speech Music Separation Using Nonnegative Matrix Factorization with Sliding Windows and Spectral Masks. INTERSPEECH, 2011

[9] J. Parras-Moral, J., F. Canadas-Quesada, P. Vera-Candeas and N. Ruiz-Reyes. 'Audio restoration of solo guitar excerpts using a excitation-filter instrument model, *Stockholm Music Acoustics Conference jointly with Sound And Music Computing Conference*, Stockholm, Sweden, 2013

[10] D. Lee and S. Seung. Learning the parts of objects by nonnegative matrix factorization, *Nature*, vol. 401, no 21, pp. 788-791, 1999

[11] D. Lee and H. Seung. Algorithms for Non-negative Matrix Factorization, in Ad- vances in NIPS, pp. 556-562, 2000.

[12] C. Fevotte, N. Bertin and J.-L. Durrieu. Nonnegative matrix factorization with the Itakura-Saito divergence. With application to music analysis. Neural Computation. 2009.

[13] http://www.zoom.co.jp

[14] Masataka Goto. Development of the RWC Music Database, Proceedings of the 18th International Congress on Acoustics (ICA 2004), pp.I-553-556, April 2004. (Invited Paper)

[15] V. Emiya, N. Bertin, B. David and R. Badeau. A piano database for multipitch estimation and automatic transcription of music. 2010.

[16] E. Vincent, R. Gribonval, and C. Fevotte. Performance measurement in blind audio source separation. IEEE Transactions on Audio, Speech and Language Processing, 14(4):1462-1469, 2006.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-191

Practical Defeasible Reasoning for Description Logics

Kody Moodley, Thomas Meyer and Uli Sattler
University of Manchester, United Kingdom and
UKZN/CSIR Meraka Centre for Artificial Intelligence Research, South Africa

moodleyk@cs.man.ac.uk;tmeyer@csir.co.za;sattler@cs.man.ac.uk

Abstract. The preferential approach to nonmonotonic reasoning was consolidated in depth by Krause, Lehmann and Magidor (KLM) for propositional logic in the early 90's. In recent years, there have been efforts to extend their framework to Description Logics (DLs) and a solid (though preliminary) theoretical foundation has already been established towards this aim. Despite this foundation, the generalisation of the propositional framework to DLs is not yet complete and there are multiple proposals for entailment in this context with no formal system for deciding between these. In addition, there are virtually no existing preferential reasoning implementations to speak of for DL-based ontologies. The goals of this PhD are: to place the preferential approach in context w.r.t. the alternative nonmonotonic reasoning proposals, to provide a complete generalisation of the preferential framework of KLM to the DL $\mathcal{ALC}$, provide a formal understanding of the relationships between the multiple proposals for entailment in this context, and finally, to develop an accompanying defeasible reasoning system for DL-based ontologies with performance that is suitable for use in existing ontology development settings.

Keywords. Defeasible reasoning, Description Logics, Nonmonotonic reasoning, Preferential reasoning, OWL, Protege, Exceptions

1. Introduction

The so-called *Preferential* or *KLM* approach [21] to nonmonotonic reasoning was introduced in the early 90's for propositional logic. In recent years, it has been shown that many of the desirable aspects of this approach can be generalised to certain fragments of first order logic such as the Description Logic (DL) $\mathcal{ALC}$ [15,8, 14]. This preferential generalisation to $\mathcal{ALC}$ has some attractive attributes. Firstly, it was shown to facilitate an intuitive representation of defeasible statements (defaults) [14,8]. It also allows one to draw desirable defeasible conclusions [9, Section 3] which are as satisfactory as (if not superior to) the more well-known circumscriptive approaches [6,16]. But the most attractive properties, yet, of this logic are that it has a reasoning procedure which is composed purely of classical DL decision steps [9]; the worst case computational complexity stays the same

as classical $\mathcal{ALC}$ ([15], [10, Corollary 2]) and preliminary experiments show that the performance in practice is promising [9].

Despite this progress, the generalisation of preferential reasoning to the case of $\mathcal{ALC}$ (let alone more expressive DLs) is not complete. There are still various theoretical results that have not been adapted or proven for this case and thus prevents a deeper understanding of the *ranked model* [21] semantics of preferential reasoning. The results we are interested in are those that lead to simpler reductions of preferential reasoning to classical decision steps and those that lead to gains in reasoning performance. Our hopes are that this investigation will also help to develop a deeper understanding of the relationship between defeasible KBs [8] ($\{C_1 \tilde{\sqsubseteq} D_1, C_2 \tilde{\sqsubseteq} D_2, ..., C_n \tilde{\sqsubseteq} D_n\}$) and their classical counterparts ($\{C_1 \sqsubseteq D_1, C_2 \sqsubseteq D_2, ..., C_n \sqsubseteq D_n\}$) which in turn would help in building defeasible reasoning systems and related tools that are intuitive and efficient to use for ontology development.

In terms of entailment, in the context of KLM preferential reasoning there are already several proposals. The consensus is that each of these proposals are suitable in different contexts. One of the aims of this PhD is to determine the relationships between these proposals and to develop an understanding of which applications each proposal is most suitable for. Of course, another important aim of this PhD is to compare our approach with other nonmonotonic reasoning proposals which are concerned with the same representation and reasoning issues.

We first give a preliminary introduction to preferential reasoning in DLs in Section 2. Thereafter, we mention some gaps in the theoretical understanding of preferential reasoning for DLs and the resulting barriers to developing simple and efficient algorithms thereof. The main contributions of this PhD are to address these specific issues: (*i*) to compare the preferential approach to related nonmonotonic reasoning proposals, (*ii*) to give a complete model-theoretic account of KLM-style preferential reasoning for $\mathcal{ALC}$, (*iii*) to determine the relationships between the different entailment proposals (hopefully discovering novel alternatives that are useful as well) and (*iv*) to apply the theoretical foundation in developing efficient algorithms for computing preferential inference on-demand.

2. Preferential Reasoning

In classical DLs [1], the semantics is built upon first order interpretations. These interpretations vary on the elements which appear in the interpretation domain ($\Delta^{\mathcal{I}}$) and the manner in which we assign terms (defined by an interpretation function ($\cdot^{\mathcal{I}}$)) to these elements. In the preferential context, we introduce an additional component on which the interpretations can vary. This component represents the manner in which we order the elements of the domain, using a *partial* ordering ($\prec_{\mathcal{I}}$). Interpretations with this additional component are known as *preferential* interpretations. In order to be able to *rank* the elements of our domain, we need to specify that the partial order be *modular* [8, Definition 1]. This is so that we are able to assign suitable *ranks* to elements that are incomparable in the partial order. Hence, preferential interpretations whose orderings are modular are known as *ranked* interpretations. The ordering component of a ranked interpre-

tation allows one to interpret so-called *defeasible* subsumption statements of the form $C \mathrel{\tilde{\sqsubseteq}} D$ (see Figure 1).

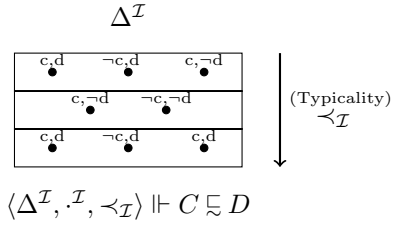


Figure 1. Satisfaction of a defeasible subsumption by a ranked interpretation.

In contrast to standard DL subsumption ($C \sqsubseteq D$), which we read as "all C's are D's", the corresponding defeasible subsumption ($C \mathrel{\tilde{\sqsubseteq}} D$) is read as "the most *typical* C's are D's". It is the ordering on the elements in a ranked interpretation that allows us to identify or specify these typical elements under consideration. The semantic paradigm which this approach captures is very intuitive because it is one which we as humans often employ (albeit in an implicit way). Consider the following example:

Example 1 *Suppose that* Bob *and* John *are mechanics. If we don't have any other information then as humans we may implicitly regard* Bob *and* John *as* typical *mechanics and assign to them properties that a typical mechanic may possess. For example we may conclude that* Bob *and* John *both work in a workshop. However, we may later discover that, while* Bob *works from a workshop,* John *is actually a* mobile *mechanic and only repairs machinery at the clients premises - which means he does not work from a workshop. One may say that* Bob *is more typical than* John *w.r.t. the property of possessing a workshop. Conversely, what this means is that* John *is more* exceptional *than* Bob *w.r.t. the same property. But what if we consider a different property of a typical mechanic? We may consider a typical mechanic to have one or more types of machinery that they* specialise *in. If we find that* John *indeed has a specialisation in motorboats but that* Bob *does not have a specialisation in any specific equipment types then we implicitly consider* John *to be more typical than* Bob *in this context.*

Example 1 demonstrates the need to consider *all* typicality orderings possible when constructing ranked interpretations of the knowledge we are reasoning about. We argue that in previous presentations of the preferential approach for DLs, there has not been enough clarity on how the approach deals with or combines *multiple* typicality orderings (as in Example 1). In Example 1 if we *only* have the constraint that typical mechanics work in a workshop then John has to be considered more exceptional than Bob in *any* ranked model thereof. Conversely, if we *only* have the constraint that typical mechanics have a specialisation then Bob is more exceptional than John. But what if we have to satisfy *both* constraints? Suppose our background knowledge is that typical mechanics work in workshops *and* that typical mechanics have at least one specialisation. Consider three of the ranked models of this knowledge in Figure 2.

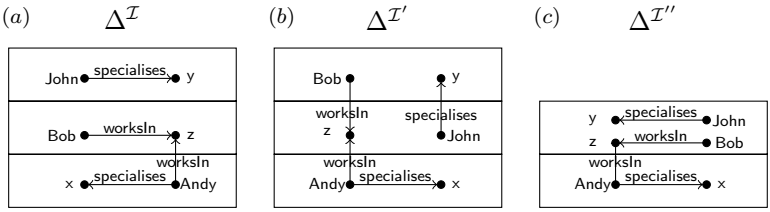


Figure 2. Combining typicality orderings using ranked interpretations.

It is clear that if our background knowledge about mechanics is correct, then there must exist at least one typical mechanic out there who satisfies both our constraints. If there isn't then we obviously have to revise or retract our statements. Since Example 1 makes mention only of Bob and John, and both these individuals are missing one of the required properties, we have to conclude that there must be a third individual. We call him Andy and he is a very typical mechanic i.e. he possesses both required properties by working in a workshop and specialising in automobiles. Both Bob and John can then be seen as exceptional w.r.t. the prototypical mechanic Andy. But how do we decide who is more exceptional between Bob and John? The answer is that we don't have to because Andy satisfies our knowledge; Bob and John are exceptional to Andy so the exceptionality distinction between them does not matter ((a), (b) and (c) in Figure 2 are all valid models of our knowledge).

A strong advantage of preferential logics is the behaviour represented in Figures 1 and 2 where the ranked interpretations satisfy that the most typical C's (lowest in the ordering) are also D's, but still allows some C's that are not as typical (higher up in the ordering) to not be D's. This is the ability to gracefully cope with *exceptions* - which is something that standard DLs cannot. We find in many fields such as biology and medicine that it is very common to encounter information which holds in general but is fallible under exceptional circumstances. Given this setting, biologists and medical professionals still have to draw conclusions and make decisions based upon this information. Preferential DLs are developed for applications of this kind.

The state of the art within this framework of ranked interpretations is that we are able to reason with the terminological part of a *defeasible KB* [9] i.e. not yet with ABoxes. A defeasible KB is composed of a classical $\mathcal{ALC}$ TBox $\mathcal{T}$ and an $\mathcal{ALC}$ *DBox* $\mathcal{D}$ (set of defeasible inclusions of the form $C \sqsubseteq D$).

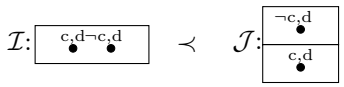
Given a defeasible KB $\langle \mathcal{T}, \mathcal{D} \rangle$, the obvious first proposal for entailment of a defeasible inclusion $C \sqsubseteq D$ would be to check in every ranked interpretation that satisfies every axiom in $\mathcal{T}$ and $\mathcal{D}$ and verify if $C \sqsubseteq D$ is also satisfied there (a similar approach is used for entailment in standard DLs). However, it turns out that this proposal induces an entailment relation which is *monotonic* [7, Section 4] and defeats the purpose of our logic, which is supposed to enable the representation of potentially fallible statements that can be refuted upon the discovery of new information.

But, even though the proposal to consider *all* ranked models fails as mentioned above, it is still possible to narrow our view to a subset of these. The prob-

lem is that deciding which subset to focus on may be perceived as a subjective choice. Fortunately, in the context of propositional logic, KLM have argued extensively that it is not entirely subjective [21,19]. They delineated a series of logical properties that any nonmonotonic consequence relation should satisfy at bare-minimum [21, Section 2.2]. They also pinpointed the smallest relation satisfying these properties coined the *Rational Closure* (RC) [21, Section 5].

A model-theoretic account of RC was also given by them which corresponds to considering the *minimal* ranked models [21, Section 5.7] as the base proposal for entailment. Minimal ranked models are defined by placing a partial ordering on the ranked models of the KB - this is in *addition* to the partial ordering on the elements of the domain (see Figure 3 for an example).

$$\langle \mathcal{T}, \mathcal{D} \rangle = \langle \emptyset, \{C \sqsubseteq D\} \rangle$$



$\mathcal{I}$ is a minimal ranked model for $\langle \mathcal{T}, \mathcal{D} \rangle$

Figure 3. Ordering ranked models in pursuit of the minimal ones.

The minimal ranked models in the partial order are those in which there is no element of the domain that can be moved to a more typical level in the strata (i.e. if it can be moved, then it is not possible without violating at least one axiom in the KB).

The logical properties that any nonmonotonic consequence relation should satisfy were shown to generalise well to the DL case ([7, Definition 4] and [8, Definition 2]). Several DL generalisations of RC have also been proposed [10,14,8,7]. Giordano et al. [14] gave the first generalisation of RC which corresponds in a natural way to the minimal ranked model semantics of KLM [21]. Our characterisation [7] was also shown to correspond to theirs.

The first attempt at a procedure for computing RC in the DL case was the effort of Casini and Straccia [10] for $\mathcal{ALC}$. This syntactic procedure was composed entirely of classical DL decision steps. A tableau calculus was presented for a preferential extension of $\mathcal{ALC}$ by Giordano et al. [15]. Notwithstanding, all existing procedures in the literature that are based on classical DL decision steps are variants of the syntactic procedure by Casini and Straccia [10].

The full technical details of our procedure including pseudocode has been presented [9]. We conclude our brief survey of preferential reasoning in DLs with an example illustrating the kinds of inferences we can draw with RC, the limitations of RC (the inferences that we would like to draw but cannot), and the additional inferences we can draw from recent extensions of RC such as the Lexicographic [20,11] and Relevant closures (submitted work).

Example 2 *Consider the following defeasible KB:*

$$\mathcal{T} = \left\{ \begin{array}{ll} 1.\ \mathsf{MRBCell} & \sqsubseteq \mathsf{ECell}, \\ 2.\ \mathsf{HRBCell} & \sqsubseteq \mathsf{MRBCell}, \\ 3.\ \mathsf{CamelRBCell} & \sqsubseteq \mathsf{MRBCell}, \\ 4.\ \exists hasShape.\mathsf{Circle} & \sqsubseteq \neg\exists hasShape.\mathsf{Oval} \end{array} \right\}$$

$$\mathcal{D} = \left\{ \begin{array}{ll} 1.\ \mathsf{ECell} & \tilde{\sqsubseteq}\ \exists hasNucleus.\top, \\ 2.\ \mathsf{MRBCell} & \tilde{\sqsubseteq}\ \neg\exists hasNucleus.\top, \\ 3.\ \mathsf{MRBCell} & \tilde{\sqsubseteq}\ \exists hasShape.\mathsf{Circle}, \\ 4.\ \mathsf{HRBCell} \sqcap \exists hasCondition.\mathsf{EMH} & \tilde{\sqsubseteq}\ \exists hasNucleus.\top, \\ 5.\ \mathsf{CamelRBCell} & \tilde{\sqsubseteq}\ \exists hasShape.\mathsf{Oval} \end{array} \right\}$$

The KB consisting of $\mathcal{T}$ *and* $\mathcal{D}$ *above represents biological information describing that: eukaryotic cells (*ECell*) usually have a nucleus, mammalian red blood cells (*MRBCell*) are types of eukaryotic cells that usually don't possess a nucleus, human red blood cells (*HRBCell*) are also mammalian red blood cells but if they are affected by the* extramedullary hematopoiesis *[25] (*EMH*) medical condition then they usually contain a nucleus. In addition to the properties pertaining to nuclei, we also represent that mammalian red blood cells generally have a circular shape but the red blood cells of a camel (*CamelRBCell*), which are also mammalian, do not inherit this property (they are distinctly oval shaped) [22].*

Using RC we are able to derive (retain) the intuitive inferences that: eukaryotic cells usually have a nucleus and even though mammalian red blood cells are considered eukaryotic, they are allowed to break the tradition and be devoid of a nucleus. In essence, mammalian red blood cells are recognised by RC as exceptional *eukaryotic cells. RC also caters for* exceptions to exceptions *by noting that a human red blood cell that is infected with* EMH *is an exceptional mammalian red blood cell and is therefore allowed to possess a nucleus.*

However, a limitation of RC is that it will not draw the reasonable inference that: human red blood cells (even if they are infected with EMH*) should be circular in shape [20,11]. We can argue that this inference is reasonable to make because we know that mammalian red blood cells usually have a circular shape (Axiom* 3 *in* $\mathcal{D}$*), and that human red blood cells are mammalian (Axiom* 2 *in* $\mathcal{T}$*). The trouble is that RC sees human red blood cells with* EMH *as exceptional even though the reason for this has nothing to do with its shape (the reason is related to the property of possessing a nucleus). Together with the fact that RC does not permit inheritance of properties for exceptional elements, the desired inference is not allowed. In an analogous way, we cannot derive another desirable conclusion that a camel red blood cell should not possess a nucleus.*

The Lexicographic and Relevant closures are syntax dependent *extensions of RC that overcome the above limitations [20,11]. They do this by identifying the reasons for information to be considered exceptional in the KB (albeit in different ways). Relevant closure (submitted work) notably uses the notion of justifications [17,4] in this regard which further exploits the connection between nonmonotonic reasoning and belief revision [13]. In both these proposals, we are able to*

derive from Example 2 that human red blood cells infected with EMH *are usually circular in shape and that camel red blood cells usually lack a nucleus.* □

3. Open Issues

As mentioned earlier, the framework for preferential reasoning in DLs is not complete. "Lifting" the theoretical results from the propositional case to the DL case is not straightforward in all situations. For example, there is a definition in the propositional case for the exceptionality of a formula w.r.t. to a defeasible KB [21, Definition 2.20]. We give the natural translation of this definition for $\mathcal{ALC}$:

Definition 1 *We define an $\mathcal{ALC}$ concept, C, as being* exceptional *w.r.t. a defeasible $\mathcal{ALC}$ KB $\langle \mathcal{T}, \mathcal{D} \rangle$ if $\langle \mathcal{T}, \mathcal{D} \rangle \models_p \top \tilde{\sqsubseteq} \neg C$. Each $C \tilde{\sqsubseteq} D \in \mathcal{D}$ is also said to be* exceptional *w.r.t., $\langle \mathcal{T}, \mathcal{D} \rangle$.*

Definition 1 uses ranked entailment ($\models_p$) to define the exceptionality of a concept C. Semantically, it means that C is considered exceptional w.r.t. the KB if the most typical elements of the domain cannot belong to the extension of C in any ranked model of the KB. In other words, from the information in the KB, it is abnormal to belong to the extension of C. This definition is quite straightforward and intuitive to understand in terms of ranked models but there is no relationship drawn to help us understand this in terms of classical DL interpretations.

Clearly, such a relationship must exist because Casini and Straccia [10] have shown that RC can be computed using purely classical DL decision steps. However, this relationship is not clearly demonstrated from a model-theoretic perspective. We argue that a reduction of the notion of exceptionality (Definition 1) to some form of classical entailment would be invaluable in demonstrating this relationship. We claim that this reduction would also help with other aspects, such as, deepening our understanding of the relationship between defeasible KBs and their corresponding classical counterparts and developing more optimised algorithms for preferential reasoning that take advantage of existing classical DL reasoner implementations.

Another issue that needs addressing is the fact that there are several alternatives to answer the question of entailment. Rational Closure, is deemed the appropriate starting point since it is the most conservative relation satisfying KLMs logical postulates [21, Section 2.2]. But for the growing number of alternatives to RC including Lexicographic Closure, Relevant Closure and even nonmonotonic reasoning proposals outside the KLM framework, there needs to be an investigation into exactly how they relate to RC and the logical postulates. We plan to investigate these relationships in terms of the entailments that they give, the applications where each is most suitable and their reasoning performance.

It is important to stress here the type of nonmonotonic formalism we are developing. The formalism is addressing the problem of classical logics being intolerant to exceptions. We wish to provide the capability of expressing potentially *fallible* statements. Note that this is different from representing *uncertain* statements. A common misconception in nonmonotonic reasoning is to conflate the meaning of these terms. The sentences "there is a sixty percent *chance* of rain

tomorrow" and "it *may* rain tomorrow" represent uncertain statements because even if our probability value of sixty is correct for the first statement there is no certainty about whether it will rain tomorrow. In contrast, the sentence "students usually don't pay taxes" is a certain statement. We are certain that students usually don't pay taxes but this "general rule" can be defeated given an exceptional case (for example, a student who works).

We are aware that there are alternatives, in the DL setting, to the preferential approach for addressing this particular problem of reasoning about exceptions. The most published approaches in the literature are based on Circumscription [5, 6,24] but there are others such as Default logic [2,3] and Minimal Knowledge and Negation as Failure (MKNF) [12,18]. Another planned contribution of this PhD is a comparison of these approaches in terms of the quality (logical merit) of their entailments and their performance.

Finally, the ultimate goal of our research is to enable the practical use of preferential reasoning in ontology development settings where DLs are the main underlying formalism. The OWL (`www.w3.org/TR/owl-features`) standard and OWL-related tools for ontology development are the main targets for the introduction of preferential reasoning features. Optimisations are needed to enable on-demand reasoning in OWL tools. In addition, various new avenues for research open up when considering non-standard reasoning services in the preferential context. Tasks such as classification (computing the subsumption relationship between each pair of concept names in the ontology) and axiom pinpointing [4] cannot be translated to the preferential context in a trivial way.

4. Methodology

Exceptionality (Definition 1) is a central principle in KLM-style preferential reasoning. It is used in preferential reasoning to induce a unique a priori ordering on the "defaults" (defeasible subsumptions) in our KB, which represents the increasing levels of "exceptional" information in the KB. We plan to determine if there is a reduction of this notion to classical entailment and if so, what this reduction is. We know classical $\mathcal{ALC}$ statements such as $C \sqsubseteq D$ can be equivalently represented by $C \sqcap \neg D \sqsubseteq \bot$. An interesting result in the preferential framework is the correspondence between $C \not\sqsubseteq \bot$ and $C \sqsubseteq \bot$. Semantically, "if there are no typical C's then there are no C's" and vice versa. This is the starting point for investigation of a reduction to classical entailment of exceptionality.

Even though KLM proposed a preferential reasoning approach in their seminal work [21], they also provided a tool (KLM postulates [21, Section 2.2]), in that same work, for evaluating the quality of the inferences drawn by *any* nonmonotonic formalism. By studying and comparing nonmonotonic logics from the perspective of the consequence relations (CR) they define, one can evaluate and compare the logical merit of the conclusions drawn, avoiding overly detailed comparisons of the mechanics of these logics. For example, any nonmonotonic formalism should satisfy Rational Monotonicity: given the consequence relation ($\not\sim$) defined by some nonmonotonic logic, this property states that if we can derive β from α ($\alpha \not\sim \beta$) and we *cannot* derive $\neg\gamma$ from α ($\not\models \alpha \not\sim \neg\gamma$) then it is "safe" to

"add" γ to α and still derive β ($\alpha \wedge \gamma \mathrel{|\!\sim} \beta$). We plan to undertake this evaluation in depth for the different nonmonotonic proposals.

Within the preferential approach, a task which is of increasing importance is to determine the relationships between the different entailment proposals. We plan to investigate this both from a theoretical perspective (e.g. LC is a venturous extension of RC, the CR of LC is a superset of the one of RC) and from a practical perspective (e.g. RC is more cautious in drawing inferences therefore it is perhaps more suitable for systems in high risk domains where extreme care should be taken before making an inference).

We have shown that we can integrate defeasible subsumption into current OWL tools with surprisingly minor modifications [23] through the use of OWL annotations. In addition, we can compute preferential reasoning using classical DL decision steps and therefore, we can exploit existing highly optimised DL reasoners to perform defeasible inference. We have a preliminary Protégé plugin which demonstrates these positive points [23].

For evaluating the performance of our algorithms, the decision of what data (defeasible ontologies) to select is a non-trivial task. We have used synthetic ontologies in the past [9] which has understandably drawn criticism because of the possible biases in their generation as well as these ontologies not resembling those found in the wild. Thus, we have to develop meaningful ways of integrating defaults into existing ontologies for our results to be more significant.

5. Conclusion

We have presented a general outline of the research that will be conducted for this PhD. The ultimate goal is to enable on-demand reasoning services for DL-based ontologies that represent defeasible subsumptions in the preferential framework. Before that goal can be achieved we propose to lift the solid theoretical foundation that was established by KLM, in propositional logic, to the DL $\mathcal{ALC}$. The different entailment proposals have to be documented and placed into perspective w.r.t. each other. These need to be compared and evaluated to determine their suitability in different contexts and practical performance. Finally, we plan to optimise the proposals and implement them in a defeasible reasoning system that can be integrated into existing OWL-related tools and systems.

Acknowledgements. This work is based upon research supported by the National Research Foundation (NRF). Any opinion, findings and conclusions or recommendations expressed in this material are those of the authors and therefore the NRF do not accept any liability in regard thereto. Kody Moodley is a Commonwealth Scholar, funded by the UK government.

References

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] F. Baader and B. Hollunder. How to prefer more specific defaults in terminological default logic. In *Proc. of the International Joint Conference on Artificial Intelligence*, 1993.

[3] F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. *Journal of Automated Reasoning*, 14(1):149–180, 1995.

[4] F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. *Journal of Logic and Computation*, 20(1):5–34, 2010.

[5] P. Bonatti, C. Lutz, and F. Wolter. Description logics with circumscription. In *Proc. of Principles of Knowledge Representation and Reasoning*, volume 6, pages 400–410, 2006.

[6] P. A. Bonatti, C. Lutz, and F. Wolter. The complexity of circumscription in DLs. *Journal of Artificial Intelligence Research*, 35:717–773, 2009.

[7] K. Britz, G. Casini, T. Meyer, K. Moodley, and I. J. Varzinczak. Ordered Interpretations and Entailment for Defeasible Description Logics. Technical report, CAIR, CSIR Meraka and UKZN, South Africa, 2013.

[8] K. Britz, T. Meyer, and I. Varzinczak. Semantic foundation for preferential description logics. In *Proc. of the Australasian Joint Conference on Artificial Intelligence*, pages 491–500. Springer, 2011.

[9] G. Casini, T. Meyer, K. Moodley, and I. J. Varzinczak. Towards practical defeasible reasoning for description logics. In *Proc. of DL*, 2013.

[10] G. Casini and U. Straccia. Rational closure for defeasible description logics. In *Proc. of JELIA*, pages 77–90, 2010.

[11] G. Casini and U. Straccia. Lexicographic closure for defeasible description logics. In *Proc. of Australasian Ontology Workshop*, volume 969, 2012.

[12] F. M. Donini, D. Nardi, and R. Rosati. Description logics of minimal knowledge and negation as failure. *ACM TOCL*, 3(2):177–225, 2002.

[13] P. Gärdenfors. *Belief revision*, volume 29. Cambridge University Press, 2003.

[14] L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato. Minimal model semantics and rational closure in description logics. In *Proc. of DL*, 2013.

[15] L. Giordano, N. Olivetti, V. Gliozzi, and G. L. Pozzato. ALC + T: a preferential extension of description logics. *Fundamenta Informaticae*, 96(3):341–372, 2009.

[16] S. Grimm and P. Hitzler. A preferential tableaux calculus for circumscriptive ALCO. In *Proc. of RR*, pages 40–54, 2009.

[17] M. Horridge. *Justification based explanation in ontologies*. PhD thesis, the University of Manchester, 2011.

[18] P. Ke and U. Sattler. Next steps for description logics of minimal knowledge and negation as failure. In *Proc. of DL*, 2008.

[19] S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.

[20] D. Lehmann. Another perspective on default reasoning. *Annals of Mathematics and Artificial Intelligence*, 15:61–82, 1995.

[21] D. Lehmann and M. Magidor. What does a conditional knowledge base entail? *Artificial Intelligence*, 55(1):1–60, 1992.

[22] R. A. McPherson, W. H. Sawyer, and L. Tilley. Band 3 mobility in camelid elliptocytes: implications for erythrocyte shape. *Biochemistry*, 32(26):6696–6702, 1993.

[23] T. Meyer, K. Moodley, and U. Sattler. DIP: A defeasible-inference platform for owl ontologies. In *Proc. of DL*, 2014.

[24] Kunal Sengupta, Adila Alfa Krisnadhi, and Pascal Hitzler. Local closed world semantics: Grounded circumscription for OWL. In *Proc. of the International Semantic Web Conference*, pages 617–632. Springer, 2011.

[25] R Verani, J Olson, and J. L. Moake. Intrathoracic extramedullary hematopoiesis: report of a case in a patient with sickle-cell disease-beta-thalassemia. *American journal of clinical pathology*, 73(1):133–137, 1980.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-201

Integration of Temporal Abstraction and Dynamic Bayesian Networks for Coronary Heart Diagnosis

Kalia Orphanou [a,1], Athena Stassopoulou [b] and Elpida Keravnou [c]
[a] *Department of Computer Science, University of Cyprus, Nicosia, Cyprus*
[b] *Department of Computer Science, University of Nicosia, Nicosia, Cyprus*
[c] *Department of Electrical and Computer Engineering and Computer Science, Cyprus University of Technology, Limassol, Cyprus*

Abstract. Temporal data abstraction (TA) is a set of techniques aiming to abstract time-points into higher-level interval concepts and to detect significant trends in both low-level data and abstract concepts. Dynamic Bayesian networks (DBNs) are temporal probabilistic graphical models that model temporal processes, temporal relationships between events and state changes through time. In this paper, we propose the integration of TA methods with DBNs in the context of medical decision-support systems, by presenting an extended DBN model. More specifically, we demonstrate the derivation of temporal abstractions which are used for building the network structure. We also apply machine learning algorithms to learn the parameters of the model through data. The model is applied for diagnosis of coronary heart disease using as testbed a longitudinal dataset. The classification accuracy of our model evaluated using the evaluation metrics of Precision, Recall and F1-score, shows the effectiveness of our proposed system.

Keywords.
temporal abstraction, temporal reasoning, Dynamic Bayesian networks, medical diagnostic models, coronary heart disease

1. Introduction

Temporal abstraction (TA) and Dynamic Bayesian networks (DBNs) have been gaining interest in the research community of medical-based systems. TA [1] is a knowledge-based process which creates high-level concepts from raw data interpreted over time intervals. The derived high-level abstract concepts have proved to be helpful in various clinical tasks and domains such as therapy planning, the summarization and interpretation of patient records [2].

DBNs [3] have been proposed in the literature to incorporate the explicit or implicit representation of time. They are the most widely used temporal extension of Bayesian networks, which are graphical models representing explicitly probabilistic relationships

[1]Corresponding author: Department of Computer Science University of Cyprus P.O. Box 20537 1678 Nicosia, Cyprus; E-mail address: korfan01@cs.ucy.ac.cy

among variables. DBNs are able to model stochastic processes in discrete time and they utilize a representation of a dynamic process via a set of stochastic variables in a sequence of time slices. DBNs have many applications in medicine in tasks such as medical diagnosis, forecasting, and medical decision making [4, 5]. A detailed survey on TA and DBN applied to medicine can be found in our recent work in [6].

In this paper, we present a novel approach of integrating TA techniques with DBNs. Our recent review of the relevant literature [6] indicated that both these areas have been largely used independently of each other in clinical domains. Our proposal is that they could be effectively integrated in the context of medical decision-support systems. We apply this integration in the medical domain of coronary heart disease (CHD) using as a testbed the STULONG dataset [2]. The proposed model, called 'DBN-extended' performs a CHD diagnosis on a particular patient based on the patient's medical history.

In particular, we use temporal abstraction methodologies to extract basic abstractions (i.e. state, single trend and persistence) using the finest possible granularity. The finest granularity is the smallest time interval period during which the variable state value remains the same and it can be acquired from experts' knowledge and raw data. The derived concepts are then used for DBN model development and deployment. Learning parameters and inference algorithms are applied to the constructed model.

The paper is structured as follows. In Section 2, we provide an overview of our approach and our testbed dataset. The methodology of deriving the temporal basic abstractions is described in Section 3 and the proposed DBN-extended model is introduced in Section 4. An extensive discussion of our experiments and experimental results is given in Section 5, and we conclude in Section 6.

2. Overview

Our goal is to integrate temporal abstraction techniques with Dynamic Bayesian networks, thus the first step is to extend the DBN network so that its nodes represent basic temporal abstractions. In order to evaluate the benefits of this integration, we developed and deployed the extended model using as a benchmark dataset, the STULONG dataset which was collected from a longitudinal study of coronary heart disease prevention. Examples of CHD events are: acute coronary syndrome, myocardial infarction, angina pectoris and ischemic heart disease. The target group includes 1428 men who may have had, or not, a CHD event before the beginning of the study.

2.1. System Overview

Our approach consists of four main phases:

i) Data preprocessing and feature selection
ii) Derivation of basic temporal abstractions (state, trend and persistence TAs) from raw data
iii) Construction of the 'DBN-extended' model and
iv) Evaluation of the model

[2]The data resource is on: http://euromise.vse.cz/challenge2004/ [Date accessed: 15 May 2014]

The first main phase consists of the feature selection process and the selection of the temporal range of observations. The selected time period is the total number of years of observations based on which the temporal abstractions were generated and the total number of time slices for the DBN were selected. We base our selection of features on the domain knowledge that we acquired from a CHD expert. The selected features are either direct or indirect risk factors (RFs) of CHD. Direct RFs include age, hypertension, cigarette smoking status (current smoker or not), dyslipidemia levels (such as Total cholesterol/HDL ratio, LDL and triglycerides levels), obesity, diabetes and history features (such as past personal history and family history). Indirect RFs include medicines treating high cholesterol (taken or not), diet (if they follow any diet or not) and exercise (if they regularly exercise or not).

The key problem for model construction is the choice of the total observation period for all patients since it ranges from 1 to 24 years. In order to remove as few records as possible from the dataset, the temporal range is chosen to be 24 years. For patients whose total observation period is less than 24 years, the CHD event is considered unknown on the years beyond their observation period. The patients' health condition is assumed to remain stable during any time period that their examination results are unknown either because their total observation period is less than 24 years or they did not take any examination at the particular time period. The target group was reduced by removing records of patients with less than three years of observations since in this study we are going to focus on the temporal aspect of the data, utilizing the advantage of long-term observations. The final target group consists of 849 individuals from whom 254 had an event at some point in time during their whole monitoring examination period.

3. Basic Temporal Abstractions

Temporal abstraction techniques are divided into two categories: basic and complex TAs. In this study we are concerned with basic temporal abstractions techniques such as: states, trend and persistence. One of the assumptions used in deriving temporal abstractions (state and trend) is that the abstraction value of a variable with missing raw values at any time within the interval period, is defined to be the same as its last known value. The same applies for cases when no record is defined during the required time interval.

3.1. State TAs

The state abstractions determine the state of an individual parameter over a particular time period based on predefined categories. The state categories for the selected features (variables) are defined by clinical experts rules. For example, poor-controlled and well-controlled hypertension are state TAs of systolic and diastolic blood pressure values. The hypertension variable is defined by the 'poor-controlled' state label if the patient has a history of hypertension and his systolic or diastolic blood pressure levels are above the standard limits; and by the 'well-controlled' state label when a patient has a history of hypertension and his systolic or diastolic blood pressure levels are normal. Otherwise, it is defined by the 'no hypertension' label. Dyslipidemia is a state TA of dyslipidemia values. It is defined as 'Present' when a patient has any of the dyslipidemia values higher than the standard limits and 'Absent' otherwise. State TAs for the Age variable are de-

rived using three state categories labels: a) 'Normal' when the patient is under 50 years old, b) 'High' when the patient is between 50 and 60 years old and c) 'Very High' when the patient is over 60 years old. State TAs for the rest of the variables are derived in a similar manner. All state TAs are displayed in **Table 1**.

Table 1. State TAs variables and their state values. Variable code is the variable name in the DBN model

Variable	Variable Code	Value=1	Value=2	Value=3
Smoking	Smoking	No Smoker	Current Smoker	
Cholesterol Medicines	medCH	Taken	Not taken	
Hypertension	HT	No Hypertension	Well Controlled	Poor Controlled
Dyslipidemia	Dislipidia	Absent	Present	
Obesity	Obesity	Absent	Present	
Age	AGE	Normal	High	Very High
Diet	DIET	Following Diet	Not Following Diet	
Exercise	Exercise	Exercising	Not Exercising	

3.2. Trend TAs

Trend abstractions of a feature are generated by observing the changes between their values. Examples of trend values are: decreasing, steady and increasing. In our approach, trend abstractions of a variable are generated by comparing two or more consecutive feature values (during the interval period of 3 years) as follows: taking into consideration the trends of all the feature values of all the examinations during a particular time period interval (3 years duration - 1-3 examinations), the most frequent trend value is selected for the corresponding feature for that period. We have also used a combination of trends and state abstractions in order to define the ratio of change of a particular variable based on its state value. Trend abstraction values are:

- 'Abnormal' when the variable state value is abnormal and its trend ratio is increasing or steady
- 'AbnormalDecr' when the variable state value is abnormal and its trend ratio is decreasing
- 'NormalInc' when the variable state value is normal and its trend ratio is increasing and
- 'Normal' when the variable state value is normal and its trend ratio is decreasing or steady

The resulting trend abstractions are displayed in **Table 2**.

3.3. Persistence TAs

Persistence TA techniques derive maximal intervals for some property by applying persistence rules both backwards and forwards in time from the specific time of the given property. Such examples are Diabetes, Family History (FH) and the past personal history of a patient for a CHD event (HistoryEvent). For example, when someone was diagnosed with diabetes at time t, diabetes is present from time t and onwards. Similarly, when

Table 2. Trend TAs variables and their trend values. Variable code is the variable name in the DBN model

Variable Name	Variable Code	Value = 1	Value = 2	Value = 3	Value =4
Total Cholesterol/HDL Ratio	TCH/HDL	Abnormal	AbnormalDec	NormalInc	Normal
LDL	LDL	Abnormal	AbnormalDec	NormalInc	Normal
Triglycerides	TRIG	Abnormal	AbnormalDec	NormalInc	Normal
HDL	HDL	Increasing	Steady	Decreasing	
Total Cholesterol	TCH	Increasing	Steady	Decreasing	

someone was diagnosed with a CHD event at time t, he has a history of event from $t+1$ and onwards, thus the value of HistoryEvent variable is 'Present' from $t+1$ until the end of the monitoring process. The FH is an example of persistence TA for the whole representation time period, since its value does not change through time. The resulted persistence TAs are displayed in **Table 3**.

Table 3. Persistence TAs variables and their persistence values. Variable code is the variable name in the DBN model

Variable Name	Variable Code	Value = 1	Value = 2
Diabetes	Diabetes	Present	Absent
Past Personal History	HistoryEvent	Present	Absent
Family History	FH	Present	Absent

4. Constructing the Dynamic Bayesian Network

The construction of the extended Dynamic Bayesian network consists of two steps: i)Building the network structure (qualitative part) and ii)Learning the parameters of the network (quantitative part).

4.1. Network Structure

The network structure, as displayed in **Figure 1**, was designed by incorporating prior information elicited from medical experts and medical literature. The derived basic temporal abstractions described in Section 3 form the nodes (variables) of our DBN. The DBN framework enables us to combine all the observations of a patient as evidence and derive a probability for the hypothesis that the patient is diagnosed with a CHD, given the total evidence gathered.

The model consists of 17 variables of which 15 are observed and two are hidden (with unknown value). Hidden variables are the class variable CurrentEvent representing the diagnosis of a CHD event and the Dislipidia node. Both of these variables take two values: 'Present' and 'Absent'. Dislipidia is introduced as a common effect node of TCH/HDL, LDL and triglycerides, which are direct risk factors to the class variable, using the parent divorcing method in order to simplify the parameters estimation process [7]. The variable FH is not repeated since it was modeled only as an initial condition and it is not changing over time. It is therefore shown in the network of **Figure 1**, to be outside the temporal plate. The arcs in the network are carriers of the causal and temporal relationships among the variables. Intra-slice arcs represent the static relationships

among variables within the same time-slice whereas inter-slice arcs connect nodes between different time slices to represent the changes over time among the variables. The single digit numbers on the arcs denote the temporal delay of the influence of the cause node to the effect. For example, an arc labeled as 1 between the variables History of CHD (HistoryEvent) and itself denotes an influence that takes one time step which is reflected to the next time slice. On the other hand, the arc without label connecting the CHD risk factors (hypertension, obesity, etc.) to the CHD event, denotes a static influence at the same time slice.

The first time slice in the network represents the time period starting from the patients' entry examination and ending three years after their entry examination. This fixed three-year granularity is chosen as it is the finer granularity, because DBNs are not able to represent irregular time periods.

4.2. Learning Parameters

Having defined the structure of the network, we need to define the conditional probabilities which quantify the arcs of the network. More specifically, we need to define the prior probability for the root nodes such as: AGE, Diet, Exercise, Smoking and FH as well as the conditional probability distributions for all non-root nodes. Each table gives the conditional probability of a child node to be in each of its states (values), given all possible parent state combinations. All of the parameters are learned from data using the expectation maximization (EM) algorithm [8].

Once the network structure is defined and the network is quantified with the learned conditional probability tables, the next step is to predict the probability of the class node CurrentEvent. Each variable in the network is instantiated by the corresponding feature value. The DBN is unrolled for eight time slices in order to represent the total observation period (24 years) of all patients included in the training dataset. Then it performs inference and derives the belief in the class variable, i.e. the posterior probability of the class at t to take on each of its values given the evidence (features) observed at the previous time step $t-1$ and at the current time-step.

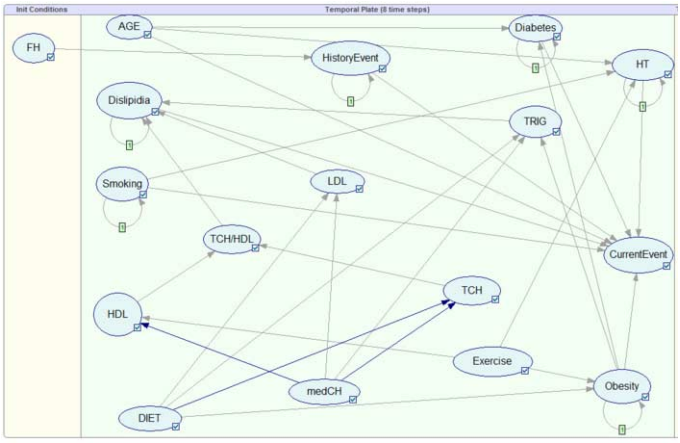


Figure 1. The graphical structure of the developed DBN model displaying the nodes on one time slice and temporal arcs representing the static or temporal relationships among variables.

5. Experimental Results and Analysis

In this section, we present the experiments performed in order to apply our methodology and evaluate the performance of our 'DBN-extended' model. For the evaluation of the accuracy of our model, we divided the dataset into training and testing using the cross-validation technique. The version of cross validation that we used in the experiments is 10 times 10-fold cross-validation, i.e. we averaged 10 runs of 10-fold cross-validation with different 10 folds in each run [9]. The classification accuracy of the model is estimated on $t = 7$ which is the last time slice.

5.1. Training in the Presence of Class Imbalance

One of the most important problems in the data mining field is to deal with imbalanced datasets. The datasets present a class imbalance when there are many more examples of one class (majority class) than of the other (minority class). It is usually the case that this latter class, i.e. the unusual class, is the class of interest. Because this unusual class is rare among the general population, the class distributions are very skewed.

In the current dataset, individuals who were not diagnosed with a CHD event at a particular time period are many more than those who were diagnosed with a CHD event (minority class). Most existing classification methods tend not to perform well on minority class examples when the dataset is extremely imbalanced. One approach to tackle the problem of an imbalanced dataset is to use resampling to modify the datasets [10, 11]. This is achieved by either removing examples from the majority class (undersampling) or adding more examples to the minority class (oversampling) or a combination of both. In our system, we evaluate our classifier on two oversampling methods as well as on a combination of oversampling with undersampling. More specifically, we apply the following resampling methods: a)SMOTE-N (Synthetic Minority Oversampling Technique for nominal features) [12], which generates synthetic examples to be added to the minority class, b)random oversampling where minority cases are randomly chosen for duplication until the ratio of majority to minority reaches a desirable level and c)SMOTE-N oversampling on the minority class with random undersampling the majority class.

We performed 4 experiments, based on resampling at various ratios. **Table 4** shows the number of patients records with a CHD event ('Present') at $t = 7$ and the number of patients records without a CHD event ('Absent') in each of the 4 training data sets:

- Dataset $D1$ is the original dataset (no resampling)
- Dataset $D2$ is defined by random oversampling the minority class
- Dataset $D3$ is obtained via oversampling using SMOTE-N and finally
- Dataset $D4$ is derived using a combination of oversampling with SMOTE-N and random undersampling.

We constructed four networks, one for each experiment. The networks had the same structure but differed in their parameters, i.e. prior probabilities and the conditional probability tables. Each time a new training dataset was introduced, new network parameters were derived using training on the new set. Throughout the remaining of the paper we will refer to the four models as: $D1$, $D2$, $D3$ and $D4$. The models presented in this paper were created and tested using the SMILE inference engine and GeNIe [3].

[3] A development environment for reasoning in graphical probabilistic models, available at: http://genie.sis.pitt.edu/. [Date accessed: 15 May 2014]

5.2. Testing the System

Two metrics that are commonly applied to imbalanced datasets to evaluate the performance of the models is recall (Eq 1) and precision (Eq 2). These two metrics are summarized into a third metric known as the F_1 measure (Eq 3). The F_1 measure is the combination of precision and recall which measures the effectiveness of classification in terms of a ratio of the weighted importance of recall and precision. In the evaluation of the proposed approach, both metrics are given equal importance. Recall and precision should be close to each other, otherwise the F_1 measure yields a value closer to the smaller of the two. Applied to our problem, precision is the ratio of the number of patients who had a CHD event at time t and are correctly classified, divided by the total number of patients who are classified of having a CHD event at time t. Recall, on the other hand, is the ratio of the number of patients who had a CHD event at time t and are correctly classified, divided by the number of patients with an actual CHD event at t.

$$Precision = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{1}$$

$$Recall = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{2}$$

$$F_1 score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3}$$

Table 4. Datasets used for four experiments with and without resampling

No of Records with Class Value:	Dataset D1	Dataset D2	Dataset D3	Dataset D4
Present (minority class)	11	55	55	55
Absent (majority class)	165	165	165	123
Total	176	220	220	178

The values of recall, precision and F_1-measure obtained from the evaluation of our model for each of the four training datasets are given in **Table 5**. As expected, the performance of the model without resampling ($D1$) is very low as this dataset is highly imbalanced and the classifier is biased towards the majority class. By applying both the random oversampling and SMOTE-N methods, we obtained dramatically improved results compared to $D1$. One risk with random oversampling, is overfitting due to placing exact duplicates of minority examples from the original set and thus making the classifier biased by remembering examples that were seen many times. The SMOTE-N technique overcomes this risk by creating synthetic examples by interpolating pairs of the closest neighbors in the minority class and introduces some new cases not included in the original dataset. The dataset derived by applying SMOTE-N oversampling combined with undersampling ($D4$) had the best classification performance. With this dataset, recall reaches as high as 91% whereas precision reaches as high as 75% yielding a combined F1-score of 82%.

Table 5. The evaluation results for all the four training datasets

Evaluation Metrics	Dataset D1	Dataset D2	Dataset D3	Dataset D4
Precision	0.176470588	0.647058824	0.680555556	0.746268657
Recall	0.272727273	0.8	0.859649123	0.909090909
F-score	0.214285714	0.715447154	0.759689922	0.819672131

We also used Receiver Operating Characteristic (ROC) curves [13, 14] to show graphically the classification performance of the four models. ROC displays graphically the trade-off between true positive rate (TPR) and false positive rate (FPR) of a classifier. TPR is the fraction of positive examples predicted correctly whereas FPR is the fraction of negative examples predicted as positive. A point on the ROC curve represents the FPR and TPR associated with the classification based on a given discrimination threshold. The threshold refers to the cut-off value above which a record is classified as positive. By varying the threshold we produce different points on the ROC curve (i.e. different (FPR,TPR) pairs). A good classification model is located as close as possible to the upper left corner of the diagram, i.e. point (TPR =1, FPR=0). The resulted graphs are displayed in **Figure 2**.

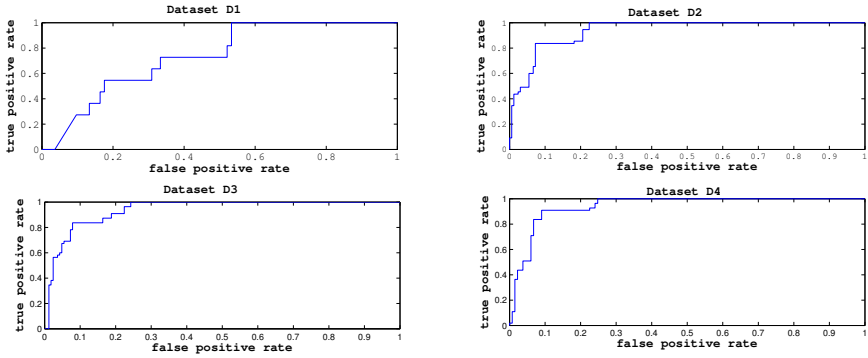


Figure 2. ROC curves $t = 7$ for all the four datasets.

6. Conclusions and Future Work

In this paper, we represented a new approach of integrating temporal abstraction with Dynamic Bayesian networks in the context of coronary heart disease. The benefits of applying our developed DBN-extended model to the CHD domain are that this extension can handle incomplete evidence in predicting disease outcomes and dealing with uncertainty which are the most usual challenges in the domain of CHD.

During our training and evaluation stages we addressed the class imbalance problem on the dataset. We have used three techniques of resampling, random oversampling, SMOTE-N and combination of SMOTE-N with undersampling to deal with the imbalance problem and developed four models by training on four different datasets. The high classification accuracy results proves the effectiveness of our proposed methodology. Our recall and precision were reaching as high as 91% and 75% respectively by apply-

ing SMOTE-N technique in combination with undersampling to the original dataset. The classification results provide a promising direction for future work. The next step is to apply the proposed approach for prognosis in the CHD domain. Estimating CHD risk for future time periods (prognosis) can help clinicians provide treatment decisions to patients that will prevent CHD events.

In addition, we are currently investigating the introduction of complex temporal abstractions to the nodes of the DBN-extended model. Complex temporal abstractions define a combination of basic TAs and/or temporal patterns. Through their representation into the DBN-extended model, we will also introduce the representation of new temporal dependencies between the variables (such as 'meets', 'overlaps' and 'starts')and the representation of events occuring at irregular time periods into the time slices.

References

[1] Y. Shahar, M. A. Musen, Knowledge-based temporal abstraction in clinical domains, Artificial intelligence in medicine 8 (3) (1996) 267–298.

[2] N. Lavrač, I. Kononenko, E. Keravnou, M. Kukar, B. Zupan, Intelligent data analysis for medical diagnosis using machine learning and temporal abstraction, AI Communications 11 (3,4) (1998) 191–218.

[3] K. P. Murphy, Dynamic bayesian networks: representation, inference and learning, Ph.D. thesis, University of California (2002).

[4] Y. Xiang, K.-L. Poh, Time-critical dynamic decision making, in: Proceedings of the Fifteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-99), Morgan Kaufmann, San Francisco, CA, 1999, pp. 688–695.

[5] T. Charitos, L. C. van der Gaag, S. Visscher, K. A. M. Schurink, P. J. F. Lucas, A dynamic bayesian network for diagnosing ventilator-associated pneumonia in ICU patients, Expert Systems Applications 36 (2) (2009) 1249–1258.

[6] K. Orphanou, A. Stassopoulou, E. Keravnou, Temporal abstraction and temporal bayesian networks in clinical domains: A survey, Artificial Intelligence in Medicine 60 (3) (2014) 133 – 149.

[7] F. V. Jensen, An introduction to Bayesian networks, Vol. 210, UCL press London, 1996.

[8] T. Moon, The expectation-maximization algorithm, Signal Processing Magazine, IEEE 13 (6) (1996) 47–60.

[9] J. M. Pena, J. Björkegren, J. Tegnér, Learning dynamic bayesian network models via cross-validation, Pattern Recognition Letters 26 (14) (2005) 2295 – 2308.

[10] A. Estabrooks, T. Jo, N. Japkowicz, A multiple resampling method for learning from imbalanced data sets, Computational Intelligence 20 (1) (2004) 18–36.

[11] A. Stassopoulou, M. D. Dikaiakos, Crawler detection: A bayesian approach, in: Proceedings of the International Conference on Internet Surveillance and Protection, ICISP '06, IEEE Computer Society, Washington, DC, USA, 2006, pp. 16–.

[12] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, SMOTE: Synthetic Minority Over-sampling Technique, Journal of Artificial Intelligence Research 16 (2002) 321–357.

[13] T. Fawcett, An introduction to ROC analysis, Pattern Recogition Letters 27 (8) (2006) 861–874.

[14] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, (First Edition), Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-211

Clause Simplifications in Search-Space Decomposition-Based SAT Solvers

Tobias PHILIPP
International Center for Computational Logic
Technische Universität Dresden, Germany

Abstract. Inprocessing is to apply clause simplification techniques during search and is a very attractive idea since it allows to apply computationally expensive techniques. In this paper we present the search space decomposition formalism SSD that models parallel SAT solvers with clause sharing and inprocessing. The main result of this paper is that the sharing model SSD is sound. In the formalism, we combine clause addition and clause elimination techniques, and it covers many SAT solvers such as PaSAT, PaMira, PMSat, MiraXT and ccc. Inprocessing is not used in these solvers, and we therefore propose a novel way how to combine clause sharing, search space decomposition and inprocessing.

Keywords. SAT, guiding paths, inprocessing, blocked clause elimination

1. Introduction

The satisfiability problem (SAT) is one of the most prominent problems in theoretical computer science and has many applications in software verification [6], planning [25, 36], bioinformatics [27] or scheduling [13]. Modern SAT solvers based on the DPLL algorithm [7] use many advanced techniques like *clause learning* [31], *non-chronological backtracking* [40], *restarts* [12], *clause removal* [3, 10], *preprocessing* [9, 28], *inprocessing* [5, 23, 39], efficient *data structures* [21, 33] and advanced *decision heuristics* [2, 3, 19, 33]. These improvements in the last decades led to a spectacular performance of conflict-driven satisfiability solvers. The *search space decomposition* approach [43] was introduced in 1996, and is a promising approach to solve *hard instances* (see [18, 32, 38] for an overview of parallel SAT solving). In particular, CCC [42] is based on this approach. The *search space*, i.e. the set of interpretations, is decomposed into different spaces by *guiding paths* [43], that are then explored in parallel by modern sequential SAT solvers. This means that the solvers compete in finding a model of the formula, and cooperate in proving its unsatisfiability.

Combining clause sharing with inprocessing in a parallel setting can make a formula unsatisfiable, as the following example demonstrates [29]: Consider the situation in which we have the two solvers $Solver_1$ and $Solver_2$ and the input formula F_0 consisting of the single unit clause (x). Then, $Solver_1$ can rewrite F_0 to its negation because this operation preserves satisfiability. Suppose that $Solver_2$ imports the negated formula, then it can terminate with the answer UNSAT because the conjunction of F_0 and its negation

is unsatisfiable, but the input formula F_0 is satisfiable. Therefore, inprocessing must be restricted if we allow clause sharing without any restriction.

To understand the interplay of advanced techniques and to reason about modern SAT solvers, we propose to formalize parallel SAT solvers. Indeed, abstracting the specific solver implementation is important since modern parallel SAT solvers consist of multiple thousand lines of code and are written in programming languages with side effects like C or C++. The formalisms presented in [1,30,34] models clause learning sequential SAT solvers, but they are inadequate to express advanced features like restarts, preprocessing, inprocessing and the ability to share clauses. But, *Generic CDCL* [20] can handle the above techniques. A formalization in [29] was introduced, that models portfolio solvers with inprocessing, and it was argued that this can be extended to guiding path solvers.

The contribution of this paper is the formal system SSD that models the computation of search-space decomposition based SAT solvers with clause sharing and inprocessing. It extends the formalism presented in [29] and we formally prove soundness of this system. We combine clause addition techniques and clause elimination techniques.

This paper is structured as follows: We start with basic notions, parallel satisfiability testing and clause simplifications in Sect. 2. Afterwards, we study some properties of guiding paths in Sect. 3, and present the sharing model SSD in Sect. 4. Finally in Sect. 5 we conclude.

2. Background

2.1. Propositional Logic

We assume a fixed infinite set $\mathcal{V}$ of Boolean *variables*. A *literal* is a variable v (*positive literal*) or a negated variable $\overline{v}$ (*negative literal*). The *complement* $\overline{x}$ of a positive (negative, resp.) literal x is the negative (positive, resp.) literal with the same variable as x. The set of all literals is denoted by $\mathcal{L}$. For a set of literals J the complement of J, denoted by $\overline{J}$, is defined as $\overline{J} = \{\overline{x} \mid x \in J\}$, and J is *consistent* if and only if $\{x,\overline{x}\} \not\subseteq J$ for every literal $x \in \mathcal{L}$. In SAT, we deal with finite clause sets called *formulas*. Each *clause* C is a finite set of literals. We write a clause $\{x_1,\ldots,x_n\}$ also as disjunction $(x_1 \vee \ldots \vee x_n)$ and a formula $\{C_1,\ldots,C_n\}$ as a conjunction $(C_1 \wedge \ldots \wedge C_n)$, where the empty clause is denoted by $\bot$.

The formula F after substituting all occurrences of the variable v with the variable w is denoted by $F[v \mapsto w]$. The set of all variables occurring in a formula F (in positive or negative literals) is denoted by $\mathsf{vars}(F)$. For a formula F and literal x, the formula consisting of all clauses in the formula F that contain the literal x is denoted by F_x.

The semantic of formulas is built on interpretations. An *interpretation* I is a set of literals that contains for all variables v exactly one of v or $\overline{v}$, and can be understood as a mapping from the set $\mathcal{V}$ of all Boolean variables to the set $\{\top,\bot\}$ of truth values. The interpretation I *satisfies* the literal x, in symbols, $I \models x$, if and only if $x \in I$. It *satisfies* the clause C, in symbols $I \models C$, if and only if there is a literal $x \in C$ such that $I \models x$. For a formula F, the interpretation I *satisfies* the formula F, in symbols $I \models F$, if and only if the interpretation I satisfies the clause C for every clause $C \in F$. A *model* I of a formula F is an interpretation I that satisfies the formula F. If there is such an interpretation I, the formula F is *satisfiable*. Otherwise, the formula F is *unsatisfiable*. The main ques-

tion can be expressed as follows: Given a formula F, the *Satisfiability Problem (SAT)* asks whether the formula F is satisfiable. Two formulas F and F' are *equisatisfiable*, in symbols $F \equiv_{\mathsf{sat}} F'$, if and only if either both are satisfiable or both are unsatisfiable. In this paper, we relate formulas by the *entailment relation*: The formula F *entails* the formula F' if and only if every model of the formula F is a model of the formula F'. This definition of entailment has two beneficial properties: transitivity and monotonicity. The drawback is that some formula simplification rules must be treated differently. Two formulas F and F' are *equivalent*, in symbols $F \equiv F'$, if and only if the formula F entails the formula F' and vice versa. The formula F' is an *unsatisfiability-preserving consequence* of a formula F if and only if $F \models F'$, and if the formula F is unsatisfiable, then the formula F' is unsatisfiable.

The *reduct of the formula F w.r.t. the consistent set of literals J*, in symbols $\mathsf{reduct}(F,J)$ is a formula obtained in two steps: First, we remove every clause from the formula F that contains a literal from J, and in the second step, we remove each literal in the remaining clauses such that the complement of the literal is contained in J.

2.2. Formula Simplifications

Simplifying the formula before giving it to the SAT solver has become an important part of the solving chain [28]. The following techniques were proposed among many others.

Subsumption Elimination The clause C *subsumes* the clause D if and only if $C \subset D$. Given an input formula $(F \wedge C \wedge D)$, where the clause C subsumes the clause D, the subsumption elimination technique produces the formula $F' = (F \wedge C)$.

Hyper Binary Resolution [4] Given an input formula F with $(y \vee x_1 \vee x_2 \vee \ldots \vee x_n) \in F$ and $(\overline{x_i} \vee z) \in F$ for $i \in \{1,\ldots,n\}$. Then, the clause $(y \vee z)$ is a *hyper binary resolvent* w.r.t. the formula F, and the hyper binary resolutions technique adds a hyper binary resolvent, i.e. the result of applying hyper binary resolution is the formula $F' = F \wedge (y \vee z)$.

Variable Elimination [9, 41] For two formulas F_1, F_2 and a variable v, the set of all non-tautological resolvents of a clause in the formula F_1 with a clause in the formula F_2 upon the variable v is denoted by $F_1 \otimes_v F_2$. Given an input formula F and a variable $v \in \mathcal{V}$, variable elimination adds all resolvents of the input formula F upon the variable v, and afterwards deletes all clauses containing the literals v or $\overline{v}$. Formally, the result of applying variable elimination is the formula $F' = (F \cup (F_v \otimes_v F_{\overline{v}})) \setminus (F_v \cup F_{\overline{v}})$.

Equivalence Elimination [17] Two variables v and w are equivalent in the formula F, if the formula F entails the equivalence $v \leftrightarrow w$. Equivalence elimination produces the formula $F' = F[v \mapsto w]$, if the variables v and w are equivalent in the formula F.

Blocked Clause Elimination [22] A clause C is called *blocked* in the formula F, if it contains a *blocked* literal x such that for all clauses $D \in F_{\overline{x}}$ the resolvent of C and D upon the variable of x is a tautology. The result of applying blocked clause elimination in the input formula $(F \wedge C)$, where the clause C is blocked in the formula F, is the formula $F' = F$.

Note that the above techniques produce unsatisfiability-preserving consequences (see [29]). Unfortunately, not all formula simplifications, like the reverse operation of blocked clause elimination [22], generate unsatisfiability-preserving consequences.

2.3. Parallel Satisfiability Solvers

Search space decomposition is an approach that distributes the search space among multiple SAT solvers that explore the assigned search spaces in parallel. The *search space* of a formula F, in symbols $\mathsf{SearchSpace}(F)$, is the set of all interpretations and the *solution space* of a formula F, denoted by $\mathsf{SolutionSpace}(F)$, is the set of all models of the formula F. Parallel solvers based on the search space decomposition approach use parallel architecture by searching in these spaces in parallel. This means that if a solver incarnation finds a model of the formula in its assigned search space, the parallel solver immediately terminates the computation and outputs SAT. Therefore, the solvers are competing w.r.t. the SAT answer. On the other hand, the solvers cooperate on finding a witness for the unsatisfiability of the formula. That means, the computation terminates with the output UNSAT, if all solver incarnations proved that no model in the corresponding search spaces exists. The dominant approach to divide the search space are *guiding paths* [43]. Guiding paths divide the search space by a conjunction of literals, and were first implemented in the parallel SAT solver PSATO [43]. A recent solver also applies *look-ahead techniques* to determine the guiding path (e.g. [42]). *Clause sharing* can be implemented in these solvers, as done in the SAT solvers PaSAT [24], PaMira [37], PMSat [11]. The SAT solver MiraXT [26] additionally performs probing-based formula simplifications during search.

2.4. State Transition Systems

State transition systems are well-understood, formal descriptions of algorithms in terms of states and binary relations over states. Formally, a *state transition system* is a tuple $(\Delta, \rightsquigarrow)$ where Δ is the set of *states* and $\rightsquigarrow \subseteq \Delta \times \Delta$ is the *state transition relation*. Given a system $(\Delta, \rightsquigarrow)$, we define $\overset{0}{\rightsquigarrow} = \{(x,x) \mid x \in \Delta\}$, $\overset{n}{\rightsquigarrow} = \{(x,z) \mid (x,y) \in \overset{n-1}{\rightsquigarrow} \text{ and } (y,z) \in \rightsquigarrow\}$ for all $n \in \mathbb{N}_{>0}$ and $\overset{*}{\rightsquigarrow} = \cup_{i \in \mathbb{N}} \overset{i}{\rightsquigarrow}$. We write $x \rightsquigarrow y$ instead of $(x,y) \in \rightsquigarrow$.

3. Guiding Paths

We start with a formal definition of guiding paths.

Definition 1 (Guiding Paths)**.** *A* guiding path *P is a finite set of non-complementary literals. A set $\mathscr{P}$ of guiding paths is* complete *if and only if for every interpretation I we find a guiding path $P \in \mathscr{P}$ such that $I \models x$ for every literal $x \in P$. The* search space *of the guiding path P is* $\mathsf{SearchSpace}(P) = \{I \mid I \models x \text{ for all literals } x \in P\}$.

The definitions here are slightly different than in [43]: In this paper a guiding path does not include information whether its assigned search space contains a model, and its assigned search space is not part of the search tree of the DPLL procedure [7, 8], but a set of interpretations.

Lemma 1 presents properties of guiding paths and the interplay of the reduct operator, guiding paths and search spaces: A complete set of guiding paths is always non-empty (Lemma 1.1), the trivial set of guiding paths $\{\emptyset\}$ is complete (Lemma 1.2). Lemma 1.3 relates complete sets of guiding paths and the search space: The union over all guiding paths of a complete set of guiding paths over the search space of P is equal

to the search space. Lemma 1.4 relates the reduct operator with the search space: The interpretation I is a model of a formula F and belongs to the search space that is defined by the guiding path P if and only if the formula $\mathsf{reduct}(F,P)$ is satisfiable. Moreover, the formula F is satisfiable if and only if for some complete set of guiding paths $\mathscr{P}$ we find a guiding path $P \in \mathscr{P}$ where $\mathsf{reduct}(F,P)$ is satisfiable (Lemma 1.5). The reduct operator is *compatible* with the entailment relation (Lemma 1.6).

Lemma 1. *Let F be a formula, I be an interpretation and $\mathscr{P}$ a set of guiding paths.*

1. *A complete set of guiding paths is non-empty.*
2. *$\{\emptyset\}$ is a complete set of guiding paths.*
3. *$\mathscr{P}$ is complete iff $\bigcup_{P\in\mathscr{P}} \mathsf{SearchSpace}(P) = \mathsf{SearchSpace}(F)$.*
4. *There is I with $I \models F$ and $I \in \mathsf{SearchSpace}(P)$ if and only if $\mathsf{reduct}(F,P)$ is satisfiable.*
5. *Let $\mathscr{P}$ be complete. Then F is satisfiable if and only if there is a guiding path $P \in \mathscr{P}$ such that $\mathsf{reduct}(F,P)$ is satisfiable.*
6. *If $F \models F'$, then $\mathsf{reduct}(F,J) \models \mathsf{reduct}(F',J)$.*

Proof. See [35]. □

4. The Search Space Decomposition Model SSD

We model the computation of SAT solvers based on the search space decomposition approach by means of a state transition system as follows: A state of computation of a single sequential solver is a formula F and guiding path P, represented as the pair (F,P). To ease our notation, we represent sequential solver states as F and say that P is the *associated guiding path*. A state of computation of a parallel SAT solver based on the search space decomposition approach of n sequential solver $Solver_1,\ldots,Solver_n$ is then an n-tuple $(F_1,\ldots,F_n)$ of formulas where F_i is the state of $Solver_i$ and P_i is the guiding path that is assigned to $Solver_i$. Then, the *set of states in* SSD *with multiplicity* n is $\{(F_1,\ldots,F_n) \mid F_i \text{ is a formula}\} \cup \{\mathsf{SAT},\mathsf{UNSAT}\}$ and we associate a guiding path P_i to the formula F_i for every $i \in \{1,\ldots,n\}$. The *initial state in* SSD *for the input formula F_0 with multiplicity n and the complete set of guiding paths* $\{P_1,\ldots,P_n\}$, denoted by $\mathsf{init}_{n,P_1,\ldots,P_n}(F_0)$, is the tuple $(F_1,\ldots,F_n)$ where the guiding path P_i is associated with the formula F_i for all $i \in \{1,\ldots,n\}$. The sharing model SSD is characterized by the following transition relation:

$$\rightsquigarrow_{\mathsf{SSD}} := \rightsquigarrow_{\mathsf{SAT}} \cup \rightsquigarrow_{\mathsf{UNSAT}} \cup \rightsquigarrow_{\mathsf{CM}} \cup \rightsquigarrow_{\mathsf{SHARE}} \cup \rightsquigarrow_{\mathsf{ADD}} \cup \rightsquigarrow_{\mathsf{DEL}} .$$

Corresponding transition rules are given in Figure 1. We have two termination rules: The SAT-rule terminates the computation in the final state SAT if and only if $\mathsf{reduct}(F_i,P_i)$ is satisfiable for some $i \in \{1,\ldots,n\}$. By Lemma 1.4 this situation happens if and only if a model of the formula F_i exists that is contained in the assigned search space of the i'th solver. Likewise, the UNSAT-rule terminates the computation in the final state UNSAT if and only if the formula $\mathsf{reduct}(F_i,P_i)$ is unsatisfiable for every $i \in \{1,\ldots,n\}$. Again by Lemma 1.4 this situation happens if and only if no model of the formula F_i exists that is contained in the search space of the i'th solver for all $i \in \{1,\ldots,n\}$. Modern complete solvers are modifying the formula by adding and removing learned clauses. Such clause

SAT-rule: $(F_1,\ldots,F_n) \rightsquigarrow_{\mathsf{SAT}} \mathsf{SAT}$
iff $\mathsf{reduct}(F_i,P_i)$ is satisfiable for some $i \in \{1,\ldots,n\}$.

UNSAT-rule: $(F_1,\ldots,F_n) \rightsquigarrow_{\mathsf{UNSAT}} \mathsf{UNSAT}$
for $\mathsf{reduct}(F_i,P_i)$ is unsatisfiable for all $i \in \{1,\ldots,n\}$.

CM-rule: $(F_1,\ldots,F_{i-1},F_i,F_{i+1},\ldots,F_n) \rightsquigarrow_{\mathsf{CM}} (F_1,\ldots F_{i-1},F_i',F_{i+1},\ldots,F_n)$
iff $F_i \equiv F_i'$.

SHARE-rule: $(F_1,\ldots,F_{i-1},F_i,F_{i+1},\ldots,F_n) \rightsquigarrow_{\mathsf{SHARE}} (F_1,\ldots,F_{i-1},F_i',F_{i+1},\ldots,F_n)$
iff $F_i' = F_i \wedge C$, and $C \in F_j$ for some $j \in \{1,\ldots,n\} \setminus \{i\}$.

ADD-rule: $(F_1,\ldots,F_n) \rightsquigarrow_{\mathsf{ADD}} (F_1 \wedge C, F_2,\ldots,F_n)$
iff $\mathsf{vars}(C) \subseteq \mathsf{vars}(F_0)$, and $F_1 \wedge C \equiv_{\mathsf{sat}} F_1$.

DEL-rule: $(F_1,\ldots,F_{i-1},F_i,F_{i+1},\ldots,F_n) \rightsquigarrow_{\mathsf{DEL}} (F_1,\ldots,F_{i-1},F_i',F_{i+1},\ldots,F_n)$
iff $i > 1$ and F_i is an unsatisfiability-preserving consequence of F_i'.

Figure 1. Transition relations used to characterize search space decomposition based SAT solvers by means of portfolio systems with input formula F_0 and multiplicity n. These definitions apply to all formulas $F_1,\ldots,F_n,F_1',\ldots,F_n'$, clauses C and $i \in \{1,\ldots,n\}$, and guiding paths P_i where P_i is the guiding path that is assigned to the solver incarnations $Solver_i$.

management is modeled by the CM-rule that rewrites a formula F_i of a solver incarnation into an equivalent formula F_i'. Finally, clause sharing is captured by the SHARE-rule that adds a clause C from the formula F_j to the formula F_i, where $i \neq j$. The ADD-rule adds a clause C to the formula F_1, if F_1 and $F_1 \wedge C$ are equisatisfiable and $\mathsf{vars}(C) \subseteq \mathsf{vars}(F_0)$. On the other hand, clause deletion is captured by the DEL-rule, which replaces a formula F_i with a formula F_i', if the F_i' is an unsatisfiability-preserving consequence of F_i, and $i > 0$.

We say that the formalism SSD is *sound* if and only if for all formulas F_0 and complete set of guiding paths $\{P_1,\ldots,P_n\}$ we have that $\mathsf{init}_{n,P_1,\ldots,P_n}(F_0) \overset{*}{\rightsquigarrow} \mathsf{SAT}$ implies that the formula F_0 is satisfiable and $\mathsf{init}_{n,P_1,\ldots,P_n}(F_0) \overset{*}{\rightsquigarrow} \mathsf{UNSAT}$ implies that the formula F is unsatisfiable. Intuitively, soundness means that every answer in the system is correct. Before we prove soundness of SSD, we establish the following invariants:

Proposition 1. *Let $n \geq 1$, let $F_0,F_1,\ldots,F_n$ be formulas, $P_1,\ldots,P_n$ be guiding paths, and let $m \geq 0$. Assume $\mathsf{init}_{n,P_1,\ldots,P_n}(F_0) \rightsquigarrow^m (F_1,\ldots,F_n)$. Then the following properties hold:*

1. $F_1 \models F_2 \wedge \ldots \wedge F_n$, *and*
2. $F_i \equiv_{\mathsf{sat}} F_0$ *for all* $i \in \{1,\ldots,n\}$.

Proof. See [29] □

We then can prove the following theorem, stating that the computed answers of the sharing model SSD are sound:

Theorem 1. *The search space decomposition model* SSD *is sound.*

Proof. We divide the proof in two parts, first proving that the output SAT is correct, then proving that the output UNSAT is correct. Let $n > 0$, and $\{P_1, \ldots, P_n\}$ be a complete set of guiding paths. Suppose $\mathsf{init}_{n,P_1,\ldots,P_n}(F_0) \overset{*}{\rightsquigarrow}_{\mathsf{SSD}} (F_1, \ldots, F_n) \rightsquigarrow_{\mathsf{SSD}} \mathsf{SAT}(\mathsf{UNSAT}, \text{resp.})$.

1. SAT: In this case $\mathsf{reduct}(F_i, P_i)$ is satisfiable for some $i \in \{1, \ldots, n\}$. Clearly, this observation implies that the formula F_i is satisfiable. By Proposition 1, we know that the formulas F_i and F_0 are equisatisfiable. Then the input formula F_0 is satisfiable and consequently the SAT answer is correct.
2. UNSAT: The proof is by contradiction: Suppose that the answer is incorrect, i.e. the input formula F_0 is satisfiable. We know by Proposition 1.2 that the formula F_1 is satisfiable. Since $\{P_1, \ldots, P_n\}$ is complete, we know by Lemma 1.5 that $\mathsf{reduct}(F_1, P_j)$ is satisfiable for some $j \in \{1, \ldots, n\}$. Hence, there is an interpretation I such that $I \models \mathsf{reduct}(F_1, P_j)$. By Proposition 1.1 we know that $F_1 \models F_2 \wedge \ldots \wedge F_n$ and conclude by Lemma 1.6 that $I \models \mathsf{reduct}(F_2 \wedge \ldots \wedge F_n, P_j)$. Since $\mathsf{reduct}(F_2 \wedge \ldots \wedge F_n, P_j) \models \mathsf{reduct}(F_i, P_j)$ for every $i \in \{2, \ldots, n\}$, we know that $I \models \mathsf{reduct}(F_i, P_j)$ for every $i \in \{2, \ldots, n\}$. Hence $\mathsf{reduct}(F_i, P_j)$ is satisfiable for every $i \in \{1, \ldots, n\}$. Then $\mathsf{reduct}(F_j, P_j)$ is satisfiable, but the UNSAT-rule is not applicable, which is a contradiction. Therefore the input formula F_0 is unsatisfiable, and consequently the UNSAT answer is correct. □

In particular, Theorem 1 states that the particular setting of clause sharing, clause addition and elimination techniques together with search space decomposition with guiding paths is sound.

5. Conclusion

Parallel SAT solvers employ many advanced techniques, but not every combination of advanced technique is sound. In particular, the combination of clause sharing and inprocessing can make a formula unsatisfiable.

Consequently, a SAT solver can incorrectly report the unsatisfiability of a formula. Fuzzying is a technique that allows SAT solver engineers to test and empirically verify combinations of techniques. One of the reasons for this methodology is the lack of tools to study such advanced combinations. Moreover, useful combinations that involve complicated constraints might be missed with the purely experimental approach. We therefore propose to formalize modern parallel SAT solvers to understand the interplay of advanced techniques, and to reason about them. In this paper, we developed a formal model in terms of state transition systems.

Search space partitioning is an early approach to use parallel hardware architectures. Guiding paths are a simple method to divide the search space among the solver incarnations. We studied guiding paths and their relation to search spaces, and afterwards developed the sharing model SSD that models parallel SAT solvers based on the guiding paths with clause sharing and inprocessing. The presented formalism covers many SAT solvers such as PASAT [24], PAMIRA [37], PMSAT [11], MIRAXT [26]. We have shown that the formalism SSD, that combines clause addition and clause elimination techniques as in [29], is sound. Therefore, the solvers can be improved by inprocessing techniques, which is a new result to the best of our knowledge.

Moreover, we indicate possibilities to improve modern parallel SAT solvers by new combinations of inprocessing and clause sharing. It is our conviction that the formalism SSD is one part of the tools desired for developing parallel SAT solvers with new combinations of advanced techniques.

As future work, we identify two interesting challenges: Biere combines restricted forms of blocked clause elimination and addition in the upcoming version of PLINGELING, and has shown empirically that it is sound. *How can we use clause addition and elimination techniques in all solver incarnations?* Moreover, the construction of unsatisfiability proofs, *DRAT* proofs [14–16], becomes an important discipline to gain confidence in UNSAT answers. *How can we create DRAT refutations for search-space decomposition based SAT solvers?*

Acknowledgement

I would like to thank Steffen Hölldobler, Norbert Manthey and Christoph Wernhard for many fruitful discussions.

References

[1] Arnold, H.: A linearized DPLL calculus with clause learning. Tech. rep., Universität Potsdam. Institut für Informatik (2009)
[2] Audemard, G., Lagniez, J.M., Mazure, B., Sais, L.: On freezing and reactivating learnt clauses. In: Sakallah, K.A., Simon, L. (eds.) SAT 2011. LNCS, vol. 6695, pp. 188–200. Springer (2011)
[3] Audemard, G., Simon, L.: Predicting learnt clauses quality in modern SAT solvers. In: Boutilier, C. (ed.) IJCAI 2009. pp. 399–404. Morgan Kaufmann Publishers Inc., Pasadena, California, USA (2009)
[4] Bacchus, F., Winter, J.: Effective preprocessing with hyper-resolution and equality reduction. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 341–355. Springer (May 2003)
[5] Biere, A.: Lingeling, Plingeling, PicoSAT and PrecoSAT at SAT Race 2010. FMV Report Series Technical Report 10/1, Johannes Kepler University, Linz, Austria (2010)
[6] Biere, A., Cimatti, A., Clarke, E.M., Fujita, M., Zhu, Y.: Symbolic model checking using SAT procedures instead of BDDs. In: Irwin, M.J. (ed.) DAC 1999. pp. 317–320. ACM (1999)
[7] Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. Commun. ACM 5(7), 394–397 (Jul 1962)
[8] Davis, M., Putnam, H.: A computing procedure for quantification theory. JACM 7(3), 201–215 (Jul 1960)
[9] Eén, N., Biere, A.: Effective preprocessing in SAT through variable and clause elimination. In: Bacchus, F., Walsh, T. (eds.) SAT 2005. LNCS, vol. 3569, pp. 61–75. Springer (2005)
[10] Eén, N., Sörensson, N.: An extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) SAT 2003. LNCS, vol. 2919, pp. 502–518. Springer (2004)
[11] Gil, L., Flores, P., Silveira, L.M.: PMSat: a parallel version of MiniSAT. JSAT 6, 71–98 (2008)
[12] Gomes, C.P., Selman, B., Crato, N., Kautz, H.: Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. Journal of Automated Reasoning 24(1–2), 67–100 (Feb 2000)
[13] Großmann, P., Hölldobler, S., Manthey, N., Nachtigall, K., Opitz, J., Steinke, P.: Solving periodic event scheduling problems with SAT. In: Jiang, H., Ding, W., Ali, M., Wu, X. (eds.) IEA/AIE 2012. LNCS, vol. 7345, pp. 166–175. Springer (2012)
[14] Heule, M., Jr., W.A.H., Wetzler, N.: Trimming while checking clausal proofs. In: FMCAD 2013. pp. 181–188. IEEE (2013)
[15] Heule, M., Jr., W.A.H., Wetzler, N.: Verifying refutations with extended resolution. In: Bonacina, M.P. (ed.) CADE 2013. LNCS, vol. 7898, pp. 345–359. Springer (2013)
[16] Heule, M., Manthey, N., Philipp, T.: Validating unsatisfiability results of clause sharing parallel SAT solvers. In: Pragmatics of SAT 2014 (2014, accepted)
[17] Heule, M.J.H., Järvisalo, M., Biere, A.: Efficient CNF simplification based on binary implication graphs. In: Büning, H.K., Zhao, X. (eds.) SAT 2011. LNCS, vol. 6695, pp. 201–215. Springer (2011)

[18] Hölldobler, S., Manthey, N., Nguyen, V., Stecklina, J., Steinke, P.: A short overview on modern parallel SAT-solvers. In: ICACSIS 2011. pp. 201–206. IEEE (2011)
[19] Huang, J.: The effect of restarts on the efficiency of clause learning. In: Veloso, M. (ed.) IJCAI 2007. pp. 2318–2323. IJCAI'07, Morgan Kaufmann Publishers Inc., Hyderabad, India (January 2007)
[20] Hölldobler, S., Manthey, N., Philipp, T., Steinke, P.: Generic CDCL – a formalization of modern propositional satisfiability solvers. In: Hölldobler, S., Malikov, A., Wernhard, C. (eds.) Proc. of the Young Scientists' International Workshop on Trends in Information Processing. vol. 1145, pp. 25–34. CEUR-WS.ORG (2014)
[21] Hölldobler, S., Manthey, N., Saptawijaya, A.: Improving resource-unaware SAT solvers. In: Fermüller, C.G., Voronkov, A. (eds.) LPAR 17. LNCS, vol. 6397, pp. 519–534. Springer (2010)
[22] Järvisalo, M., Biere, A., Heule, M.: Blocked clause elimination. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 129–144. Springer (2010)
[23] Järvisalo, M., Heule, M., Biere, A.: Inprocessing rules. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS, vol. 7364, pp. 355–370. Springer (2012)
[24] Jurkowiak, B., Li, C.M., Utard, G.: Parallelizing satz using dynamic workload balancing. Electronic Notes in Discrete Mathematics 9, 174–189 (2001)
[25] Kautz, H.A., Selman, B.: Planning as satisfiability. In: Neumann, B. (ed.) ECAI 1992. pp. 359–363 (1992)
[26] Lewis, M., Schubert, T., Becker, B.: Multithreaded SAT solving. In: ASP-DAC 2007. pp. 926–931. IEEE Computer Society, Washington, DC, USA (2007)
[27] Lynce, I., Marques-Silva, J.: SAT in bioinformatics: Making the case with haplotype inference. In: Biere, A., Gomes, C.P. (eds.) SAT 2006. LNCS, vol. 4121, pp. 136–141. Springer (2006)
[28] Manthey, N.: Coprocessor 2.0 – a flexible CNF simplifier. In: Cimatti, A., Sebastiani, R. (eds.) SAT 2012. LNCS, vol. 7317, pp. 436–441. Springer Berlin Heidelberg (2012)
[29] Manthey, N., Philipp, T., Wernhard, C.: Soundness of inprocessing in clause sharing SAT solvers. In: Järvisalo, M., Van Gelder, A. (eds.) SAT 2013. LNCS, vol. 7962, pp. 22–39. Springer (2013)
[30] Marić, F.: Formalization and implementation of modern SAT solvers. J. Autom. Reason. 43(1), 81–119 (2009)
[31] Marques Silva, J.P., Sakallah, K.A.: Grasp: A search algorithm for propositional satisfiability. IEEE Transactions on Computers 48(5), 506–521 (1999)
[32] Martins, R., Manquinho, V., Lynce, I.: An overview of parallel SAT solving. Constraints 17(3), 304–347 (2012)
[33] Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: engineering an efficient SAT solver. In: DAC 2001. pp. 530–535. Association for Computing Machinery (2001)
[34] Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). JACM 53(6), 937–977 (Nov 2006)
[35] Philipp, T.: Expressive Models for Parallel Satisfiability Solvers. Master thesis, Technische Universität Dresden (2013)
[36] Rintanen, J.: Engineering efficient planners with SAT. In: Raedt, L.D., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P.J.F. (eds.) ECAI 2012. Frontiers in Artificial Intelligence and Applications, vol. 242. IOS Press (2012)
[37] Schubert, T., Lewis, M., Becker, B.: PaMira - a parallel SAT solver with knowledge sharing. In: Abadir, M.S., Wang, L.C. (eds.) MTV 2005. pp. 29 –36 (2005)
[38] Singer, D.: Parallel Resolution of the Satisfiability Problem: A Survey, chap. 5, pp. 123–148. Wiley Interscience (2006)
[39] Soos, M.: CryptoMiniSat 2.5.0. In: SAT Race Competitive Event Booklet (July 2010)
[40] Stallman, R.M., Sussman, G.J.: Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. Artificial Intelligence 9(2), 135–196 (1977)
[41] Subbarayan, S., Pradhan, D.K.: NiVER: Non-increasing variable elimination resolution for preprocessing SAT instances. In: Hoos, H.H., Mitchell, D.G. (eds.) SAT 2004. LNCS, vol. 3542, pp. 276–291. Springer (2005)
[42] van der Tak, P., Heule, M., Biere, A.: Concurrent cube-and-conquer. In: Cimatti, A., Sebastiani, R. (eds.) SAT 2012. LNCS, vol. 7317, pp. 475–476. Springer (2012)
[43] Zhang, H., Bonacina, M.P., Hsiang, J.: PSATO: a distributed propositional prover and its application to quasigroup problems. Journal of Symbolic Computation 21(4), 543–560 (1996)

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-220

Multi-objective learning of accurate and comprehensible classifiers – a case study

Rok PILTAVER[a,b], Mitja LUŠTREK[a] and Matjaž GAMS[a,b]
[a]*Jožef Stefan Institute - Department of Intelligent Systems, Ljubljana, Slovenia*
[b]*Jožef Stefan International Postgraduate School, Ljubljana, Slovenia*

Abstract. Accuracy and comprehensibility are two important classifier properties, however they are typically conflicting. Research in the past years has shown that Pareto-based multi-objective approach for solving this problem is preferred to the traditional single-objective approach. Multi-objective learning can be represented as search that starts either from an accurate classifier and modifies it in order to produce more comprehensible classifiers (e.g. extracting rules from ANNs) or the other way around: starts from a comprehensible classifier and modifies it to produce more accurate classifiers. This paper presents a case study of applying a recent algorithm for multi-objective learning of hybrid trees MOLHC in human activity recognition domain. Advantages of MOLHC for the user and limitations of the algorithm are discussed on a number of datasets from the UCI repository.

Keywords. Multi-objective learning, hybrid classifier, hybrid tree, accuracy, comprehensibility.

Introduction

When evaluating a classifier, one is usually most interested in its predictive accuracy estimated by e.g. percent of correctly classified instances, confusion matrix, area under ROC, or other measures. However, there are also other classifier properties that are often important for the user: comprehensibility [1] – also referred to as understandability or interpretability – justifiability [2, 3], surprisingness [4], and others. This paper is limited to discussing accuracy and comprehensibility.

The comprehensibility is defined as: "the ability to understand the output of induction algorithm" [5] or "the ability to understand the logic behind a prediction of the model" [6]. According to Craven and Shavlik [7] it is important because it enables: classification explanation, classifier validation, knowledge discovery and supports classifier generalization improvement and refinement of approximately-correct domain theories. Furthermore, there are many application domains in which the importance of comprehensible classification models continues to be emphasized, such as: medicine, credit scoring, churn prediction, and bioinformatics [1].

The main problem in learning accurate and comprehensible classifiers is that the two objectives are conflicting [8]. There are two main approaches to solving this problem [8, 9]. The weighted-formula approach is conventional; it transforms the multi-objective problem into a single-objective one. The second approach is Pareto-based multi-objective approach. Its objective function is no longer a scalar value, but a vector so all the criteria are treated separately. This produces a number of Pareto-optimal solutions [10] (i.e. classifiers) instead of a single solution. Freitas [9] lists

arguments for and against each approach and concludes that the more complex Pareto-based approach is preferred because it avoids multiple runs of single-objective optimisation algorithm and the ad-hoc specification of its parameters (i.e. weights) as well as provides very informative set of non-dominated solutions [10]. Nevertheless, depending on the application domain, there are cases in which the weighted-formula approach is sufficient or the Pareto-based approach is too complex.

To learn accurate and comprehensible classifier an algorithm can start with an accurate classifier and transform it to produce more comprehensible ones. Examples of such approaches are extracting rules from artificial neural networks (ANN) [8] and pruning decision trees to find a trade-off between their size – which is related to its comprehensibility – and accuracy [11]. The search can also proceed inversely: start from a comprehensible classifier and transform it to produce more accurate classifiers. An example of such algorithm is the recently presented multi-objective learning of hybrid classifiers (MOLHC) algorithm [12], which is guaranteed to find the entire Pareto set of hybrid trees by replacing sub-trees in the initial classification tree with black-box classifiers (e.g. SVM, ANN, or random-forest).

This paper presents a case study of applying the MOLHC algorithm in human activity recognition domain: motivation for learning hybrid classifiers and the insights in the classification task provided by the visualization of the algorithm's output. Limitations and performance of MOLHC algorithm are discussed on a number of datasets from the UCI repository [13, 14].

The structure of the paper is as follows: Section 1 gives a quick overview of the MOLHC algorithm, Section 2.1 introduces the activity recognition domain used for the case study, Section 2.2 illustrates the use of the algorithm, its advantages and drawbacks, and finally Section 3 summarizes the paper and suggests directions for further research.

1. The multi-objective learning of hybrid classifiers algorithm

The basic idea of the multi-objective learning of hybrid classifiers (MOLHC) algorithm [12] is to replace sets of leaves in a given comprehensible classification tree with black-box (BB) leaves that invoke a provided accurate BB classifier in order to increase the accuracy of the resulting hybrid trees compared to the initial tree. The algorithm is motivated by the fact that many machine learning domains as well as human expert knowledge can be partially explained with simple models (e.g. rules) but require much more complex and less comprehensible model in other parts. The algorithm guaranties to find the complete Pareto set of described hybrid trees efficiently: it outperforms a state of the art multi-objective optimisation algorithm NSGA-II [17] for the discussed task in terms of run-time, and in contrast with NSGA-II guaranties finding the complete Pareto set (i.e. is not stochastic) and does not require setting any search parameters [12]. It has been shown to produce sets of hybrid trees that considerably outperform the baseline algorithms (classification tree and BB classifier) in terms of hyper-volume under the attainment surface in many domains [12]. In simple words: the set of hybrid trees produced by MOLHC algorithm consists of classifiers that cannot be constructed with the baseline algorithms and offer useful trade-off between accuracy and comprehensibility.

In contrast with most related work MOLHC does not use the size of classification tree as a measure of comprehensibility because it operates with hybrid trees. Instead,

the comprehensibility c of a hybrid tree is defined by Eq. 1 as the ratio between the number of examples that are classified by the regular leaves i and the number of all examples N used to evaluate the comprehensibility of the hybrid tree.

$$c = (\sum_{i \text{ is non-replaced leaf}} N_i)/N \quad (1)$$

The comprehensibility of a hybrid tree is therefore equal to the probability of classifying an instance with a comprehensible model. By definition, the comprehensibility of the initial classification tree is 1. A classification tree is considered as perfectly comprehensible regardless of its size; however it is only sensible to use the measure and the algorithm on reasonably small classification trees, i.e. with less than ~50 leaves. Comprehensibility of the BB classifier is 0, meaning that it is not comprehensible at all. The comprehensibility of all other hybrid trees are between 0 and 1.

A naïve approach to finding the Pareto set of hybrid trees would be to generate, evaluate and compare all the possible hybrid trees. This yields a search space with 2^n hybrid trees where n is the number of leaves in the initial classification tree that are considered to be replaced with BB leaves. Only the leaves in which BB classifier achieves higher accuracy than the majority class classifier belonging to the leaf should be considered for replacing with BB leaves - replacing leaves for BB decreases comprehensibility and is therefore only worthwhile if it increases accuracy at the same time. The MOLHC examines the search space using an iterative search methods that avoids generating most hybrid trees not belonging to the Pareto set. The main loop of the algorithm considers a set of hybrid trees that have the same number of replaced leaves. It starts with hybrid tree that has zero replaced leaves (only the initial tree belongs to this set) and increases until it has considered replacing all the leaves. When considering a hybrid tree (from the set of hybrid trees processed in the current iteration), it produces a set of new hybrid trees that can be generated from the given hybrid tree by replacing exactly one non-dominated leaf. The set of non-dominated leaves L_n is defined by Eq. 2 and contains the leaves l that are better than all the other currently non-replaced leaves L considering the difference in accuracy a_l and comprehensibility c_l introduced by replacing the leaf for the BB – they increase accuracy more and at the same time decrease comprehensibility less than other leaves.

$$L_n = \{\, l \in L;\ \forall i \in L: (a_l \geq a_i) \wedge (c_l \leq c_i) \,\} \quad (2)$$

Replacing only the non-dominated leaves limits the search but has been proven to find the complete Pareto set of hybrid trees [12]. The search optimisation enables exact multi-objective learning based on initial trees with less than ~50 leaves in under a second on a personal computer (e.g. 3 GHz Intel® Core™ 2 Duo) [12]. For comparison consider that a naïve algorithm takes over 3 minutes for an initial tree with 17 leaves and 18 minutes for 18 leaves; bigger trees cannot be used with the naïve algorithm as its time complexity increases exponentially with the number of leaves [12].

2. Case study of multi-objective learning of hybrid classifiers algorithm

In order to demonstrate the use of MOLHC algorithm in practice and show the advantages of multi-objective learning and hybrid trees with black-box (BB) leaves this section presents a case study of MOLHC application in activity recognition domain described in the following subsection. In addition we selected the datasets for testing from the set of 94 classification datasets from the UCI repository [13] available in ARFF format at the Weka webpage [14]. Among the 49 datasets with more than 300 instances we chose 23 datasets where the BB classifier achieved at least 10 % better accuracy than the tree with approximately 20 leaves. Finally 40 trees were used to calculate the results shown in Figure 2 and Figure 4: one small tree (~20 leaves) for each of the 23 datasets and another bigger tree (~40 leaves) for 17 dataset that allowed building larger trees. The choice of datasets and initial trees is the same as in [12].

2.1. Motivation for MOLHC application and the case study domain

The goal of activity recognition is to recognize the activity a person is performing using sensors and software for sensor data processing. The task can be to recognize: *basic activities* such as lying, sitting, standing, walking, running, cycling, transitions between activities, etc.; *events* such as falling, sitting down, standing up, stop moving, etc.; or *complex activities* such as performing house chores, preparing meal, eating, exercising, shopping, etc. Sensor data can be obtained from video cameras, real-time locating systems, inertial sensors, or 3D motion capture systems. Activity recognition is an important task in ambient intelligence as it is prerequisite for many applications such as sport applications, health monitoring, smart house automation, and others. This section considers learning a classifier that distinguishes between 10 basic activities (listed in the following paragraph) based on attributes extracted from data provided by a single 3-axis accelerometer mounted on person's chest. The recognized basic activities can then be used to recognize events and complex activities.

The training and testing dataset was recorded in a laboratory with 9 persons each performing a given sequence of activities lasting for approximately 1.5 hours: 22 % of the time lying, 17 % walking, 14 % cycling, 10 % standing, 7-8 % of sitting, kneeling, on all fours, and running (each), 4 % transitions between activities, and 3 % leaning. Two-second time windows of measured accelerations along each of the 3 accelerometer axes was used to compute 61 attributes suggested by literature, for example: mean value, area under the curve, amplitude, total energy, dominant frequency, mean crossing rate, entropy, variation coefficient, etc. The time windows were overlapping (1 second overlap with the previous window and 1 second overlap with the following window) therefore a total of around 48.000 instances were acquired.

We intended to enter the EvAAL live activity-recognition competition (http://evaal.aaloa.org/) therefore we required a classifier that we could trust to perform correctly in a situation substantially different from the one in the laboratory. The sequence and duration of activities that would be used for testing at the competition were not known. In addition the placement of the accelerometer could not be guaranteed to be exactly the same as in the laboratory and the motion of the person evaluating the activity recognition system at the competition could be different then the motion recorded in the laboratory, e.g. different posture or intensity of movement. A very accurate (90.6 % classification accuracy) black-box classifier was constructed using high-quality laboratory data, however it did not allow an expert to validate it. On

the other hand, completely comprehensible classifiers performed poorly (77.0 % classification accuracy) in comparison. Hence this was a problem that called for a hybrid approach using MOLHC to generate a set of hybrid classifiers ranging from the most comprehensible to the most accurate in order to get an insight in the classification problem at hand and choose a classifier (hybrid tree) that has high enough accuracy and is as comprehensible as possible.

2.2. MOLHC for activity recognition

The first step in using MOLHC is to **choose an initial classification tree** and a black-box (BB) classifier with high accuracy. First a classification tree was constructed using the original dataset but was difficult to validate. In order to support validation the domain expert choose a subset of 12 attributes that were known to be important for the classification and were easy to interpret. They were used to build a classification tree using C4.5 learning algorithm [15] implemented as J48 in Weka [16]. Its pruning parameters were set so that it produced a tree of appropriate size (12 leaves) and accuracy (76.1 %). The tree is shown in Figure 3 – attributes are renamed and numerical attribute values that split the data into sub-trees are replaced with words in order to improve comprehensibility for readers not familiar with the domain. The size of the initial tree should be small enough to prevent overfitting and enable the expert to analyse it in reasonable time. On the other hand pruning a tree too much will decrease its accuracy. The initial classification tree can also be built by a domain expert based on his knowledge: it should include the rules he knows are valid and possibly some additional rules he suspects are valid in most cases.

To **choose a BB classifier**, several classifiers should be trained using various learning algorithms and learning parameter settings, and compared according to their accuracy estimated on a test set. In our case, random forest classifier was chosen as the BB classifier (90.6 % classification accuracy) based on the experts' past experience with the domain. All the 61 available attributes were used for learning the random forest classifier – there is no point in holding back any data (except redundant and random attributes) from the algorithm that learns the BB classifier. A possible improvement of the algorithm could use multiple BB classifiers: one for each leaf or one for each sub-tree with enough examples to learn an accurate BB classifier.

The second step is the **execution of the MOLHC algorithm,** which uses the following inputs: the initial classification tree, BB classifier, and data that was not used to train the two input classifiers. The output of the algorithm is a Pareto set of hybrid trees, which is represented in the objective space as the Pareto front: a graph with accuracy on one axis, comprehensibility on the other, and points on the graph representing individual hybrid trees (see Figure 1). By analysing the Pareto front and data about the Pareto set, knowledge about the domain can be extracted as illustrated in the following paragraphs.

The first thing to observe is the **steepness of the Pareto front**. A steep Pareto front (e.g. Figure 1a) represents a case in which the difference in accuracy between the initial tree and BB classifier is small. In such cases the user should consider choosing the initial tree because it achieves accuracy similar to the BB classifier but is completely comprehensible as opposed to the BB. On the other hand, a Pareto front that decreases gradually (e.g. Figure 1c) represents a case with considerable difference in accuracy between the initial tree and BB classifier. In such cases the user should investigate the Pareto front further in order to select an appropriate hybrid tree with a

desired trade-off between accuracy and comprehensibility. The steepness of Pareto front corresponds to the difficulty of classification task for a decision tree (given a comprehensible decision tree and an accurate BB classifier). Figure 1 shows that iris data set [13] is easy to classify using a comprehensible classifier, while a simple classifier does not suffice to classify activity or letter datasets [13] with high accuracy.

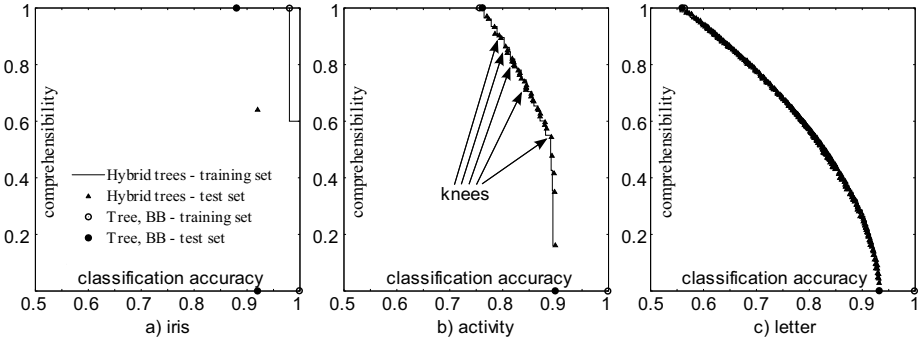


Figure 1. Pareto fronts of hybrid trees for a) iris, b) activity, and c) letter datasets

The second property of the Pareto front to be investigated is the **density and spread of hybrid trees along the Pareto front**. For instance the Pareto front for iris dataset (Figure 1a) is sparse (includes only two hybrid trees), while the Pareto front for letter dataset (Figure 1c) is dense (403 hybrid trees). Deb [10] lists several measures of spread, however a threshold between sparse and dense Pareto front depends on the application domain. If there are few hybrid trees on a Pareto front the user can inspect and compare all of them, otherwise he should concentrate on a subset of hybrid trees. Pareto front in Figure 1c is well spread along the entire range while the Pareto front in Figure 1b is considerably sparser in the low comprehensibility range. Hybrid trees in the sparse part of the Pareto front should be inspected thoroughly while only a subset of trees needs to be inspected in the dense part since it contains many similar trees – the similarities usually become obvious quickly, however a program with appropriate graphical interface could automate and simplify the task further. Figure 2 shows that the number of hybrid trees and hence the density of Pareto front depends on the number of leaves in the initial tree for which the accuracy of BB classifier is higher than the accuracy of majority class classifier in the leaf. It also depends on the differences in accuracy and comprehensibility that is introduced by replacing each leaf for a BB leaf, which explains the outliers in Figure 2.

The third important property of the Pareto front is **presence of knees**: parts of Pareto front with sudden jump in one of the objectives. Jin [18] argues that quantitative measure for knee should be defined according to the application domain. Example of a knees are shown in Figure 1b; the most obvious one occurs where accuracy approaches 0.9. Knees are important because they limit the set of hybrid trees that needs to be examined by the user. For instance, if a hybrid tree with high accuracy is requested in the activity dataset (Figure 1b) then the one with accuracy 0.89 and comprehensibility 0.55 is a good candidate (an arrow pointing at it in Figure 1b). It has almost the same accuracy as the hybrid trees further down the Pareto front but it has considerably higher accuracy. Among the hybrid trees near a knee, the ones that have extreme values of an objective are the most interesting for the user.

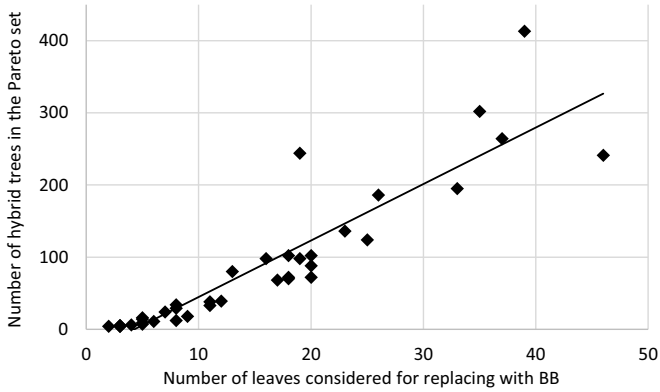


Figure 2. Number of hybrid trees in a Pareto set depends on the number of leaves in the initial tree that are considered for replacing with BB leaves (results obtained from 40 initial trees built on 23 UCI datasets).

Another insight offered by the MOLHC approach is **validation of classification tree leaves**. It is most useful if the initial tree is constructed by an expert (not by a machine learning algorithm) as it validates the experts knowledge and exposes expert's assumptions that are not in line with the provided data; it can be used to validate a learned tree as well. Among the 12 leaves in the initial tree (used for the activity recognition domain) BB achieved higher accuracy in all but the leaf number 8 (Figure 3). Because the BB classifier cannot improve classification accuracy for the instances belonging to that leaf, the user can accept the leaf as valid peace of extracted knowledge. For iris dataset, which can be accurately classified with a classification tree, there were three leaves in the initial tree and BB classifier outperformed them in only one leaf while the other two were confirmed as valid.

Besides checking in which leaves BB classifier achieves higher accuracy then the leaf, the Pareto set of hybrid classifiers is analysed in order to calculate the **relative quality of leaves**. The algorithm counts the number of Pareto optimal hybrid trees in which a leaf was replaced for a BB leaf. If the count for a leaf is low, it means that the leaf is good according to both objectives (accuracy and comprehensibility): it correctly classifies a large share of instances belonging to the leaf and provides classification explanation for a big number of instances. Discretized counts are depicted as stars under each leaf in Figure 3: a high number of black stars represents leaves with good accuracy and comprehensibility and vice versa. Figure 3 shows that leaves with high probability of the class assigned in the leaf (percent of instances belonging to the class are given in each leaf) receive good score, which is to be expected. However, these scores provide additional information: leaves 7 and 10 have similar classification accuracy (~56 %), but leaf 10 has lower score. This means that BB classifier is able to improve the classification accuracy in leaf 10, but not in leaf 7. A quick look at the number of stars in Figure 3 revels that running, walking, cycling and lying activities are easy to recognize while sitting, kneeling and standing are often confused by the classification tree and are classified with low accuracy. Therefore, they should be replaced by BB classifier in order to improve accuracy, since comprehensible classification is not provided by the initial tree. The domain expert confirmed that additional accelerometer on thigh should be used in order to distinguish sitting from standing. He also confirmed that the suggested four activities are easy to classify: an

older version of activity recognition software that he developed used hand crafted rules, similar to the ones in the tree, to recognize the four activities. This illustrates how the MOLHC supports the knowledge discovery and classifier validation.

The domain expert finally **choose the hybrid tree** (Figure 3), which achieves 84.1 % accuracy and comprehensibility 0.72. By sacrificing some accuracy he obtained an accurate classifier enabling good classifier validation. The chosen hybrid tree replaced a sub-tree (containing leaves 1-3) with a single BB leaf, which increased the overall accuracy by 3.3 % and decreased comprehensibility by 0.13. It also replaced leaves 5 and 10, which increased accuracy by additional 4.7 % and decreased comprehensibility by additional 0.15. The expert could change the initial tree by adding sub-trees to those two leaves and run the MOLHC again if higher comprehensibility and similar accuracy was required. This illustrates how MOLHC supports classifier generalization improvement and refrainment of approximately-correct domain theories.

Since the user chooses a hybrid tree based on the Pareto front, it is very important that the comprehensibility and accuracy values used for drawing the Pareto front are accurate. They are estimated on the training set and therefore depend on the number of training instances as is shown in Figure 4. Insufficient number of training instances may lead to **errors in the estimated comprehensibility and accuracy** and therefore mislead the user while choosing a hybrid tree. Figure 1a shows one such example, which occurred because only 50 instances were used for the iris dataset. The problem is only amplified by the fact that MOLHC approach requires three datasets: one for learning the initial tree and BB classifier, another for learning the Pareto set of hybrid trees and usually also the third for evaluating the hybrid trees. An improvement of the algorithm, which would enable it to perform reliably with small datasets and would limit the errors of the predicted comprehensibility and accuracy of the hybrid trees, would be welcome. It could probably be achieve using internal n-fold cross-validation.

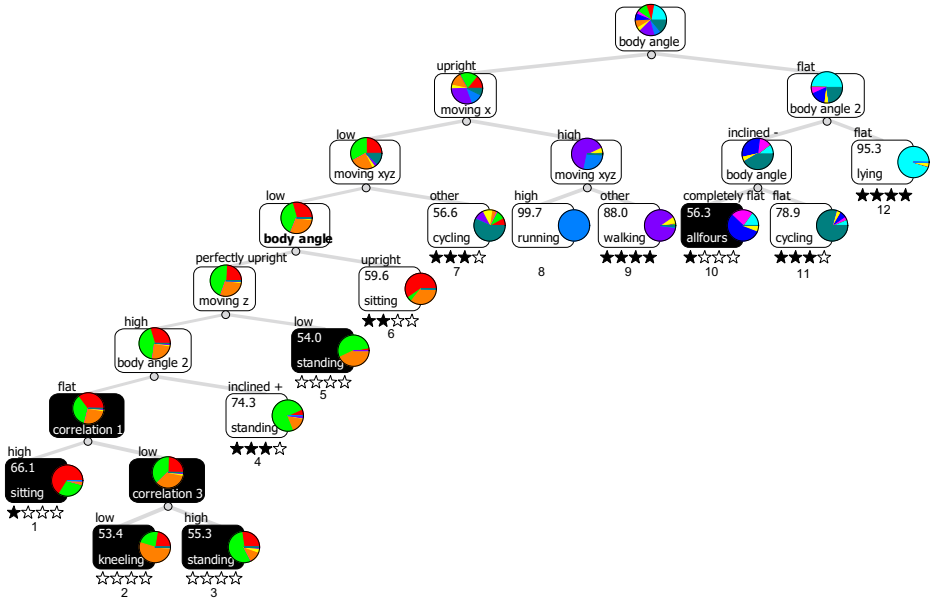


Figure 3. Output of the MOLHC algorithm for the activity recognition domain: quality of the leaves (stars), black-box leaves of the chosen hybrid tree (black leaves), and pie charts representing class distributions in each node.

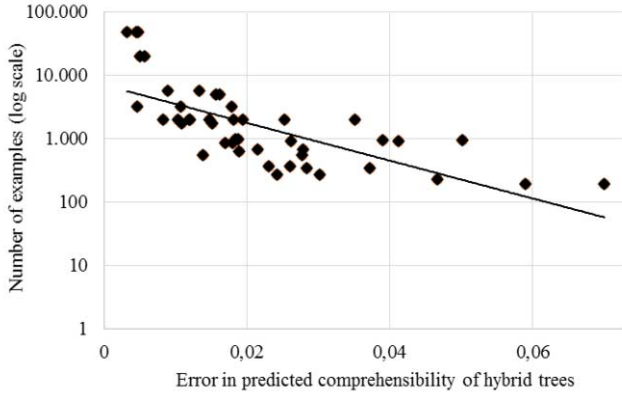


Figure 4. Error in predicted comprehensibility of hybrid trees depends on the number of learning examples (results obtained from 40 initial trees built on 23 UCI datasets).

3. Conclusion

This paper presents the motivation for and advantages of using multi-objective learning algorithm MOLHC in a real world use case. The algorithm graphically presents the difficulty of classification task for a comprehensible classifier, identifies parts of the domain that can be classified accurately with understandable classifier, and parts of the domain that are more challenging and should be classified with a black-box (BB) classifier instead. It offers an insight into the analysed classification problem and supports classifier validation, knowledge discovery, and refinement and improvement of classifiers, which are important features according to [7]. The output of the algorithm is the Pareto set of hybrid trees, which range from most comprehensible to the most accurate. The paper shows that the size of the Pareto set depends mostly on the number of leaves in initial tree in which BB classifier achieves higher accuracy than the majority class classifier of the leaf. The Pareto front supports the user in taking well informed decision when choosing a hybrid tree that should be both accurate and comprehensible. Furthermore, the paper shows that the error of predicted comprehensibility depends on the number of learning instances, which could be a limiting factor for MOLHC application on domains with few instances. Another drawback of using MOLHC is a possible large number of hybrid trees presented on the Pareto front; however, the case study shows that the user needs to focus only on a subset of those hybrid trees that satisfies the requirements about accuracy and comprehensibility. Furthermore, presence of knees on the Pareto front and scoring of leaf quality further decreases the number of hybrid trees that must be compared by the user.

Future work should be devoted to decreasing the error of predicted comprehensibility and accuracy of hybrid trees and enabling reliable algorithm performance on small datasets. Using multiple BB classifiers should be investigated as it could provide improvements in accuracy of the hybrid trees. A program with appropriate graphical user interface could improve the user experience with the MOLHC algorithm and provide additional insights into the classification problem.

Systematic investigation of exploiting the Pareto set of hybrid trees to calculate the qualities of individual leaves also seems promising.

References

[1] A. A. Freitas, Comprehensible classification models - a position paper. *ACM SIGKDD Explorations*, vol 15-1 (2013), 1-10.

[2] H. Allahyari, N. Lavesson, User-oriented Assessment of Classification Model Understandability, *in Proceedings of the Eleventh Scandinavian Conference on Artificial Intelligence,* (2011), 11-19, IOS Press, ISBN 978-1-60750-753-6

[3] D. Martens, B. Baesens, Building Acceptable Classification Models, Data Mining - *Annals of Information Systems*, vol 8 (2010), 53-74

[4] D. R. Carvalho, A. A. Freitas, N. F. F. Ebecken, A Critical Review of Rule Surprisingness Measures, *in Proceedings of Data Mining IV - International Conference on Data Mining*, (2003) 545-556.

[5] R. Kohavi, Scaling Up the Accurcy of Naive-Bayes Classifiers: a Decision-Tree Hybrid, *Second International Conference on Knowledge Discovery and Data Mining,* (1996), 202-207, AAAI Press.

[6] D. Martens , J. Vanthienen, W. Verbeke, B. Baesens, Performance of classification models from a user perspective, *Decision Support Systems*, vol 51-4 (2011), 782–793.

[7] M. W. Craven, J. W. Shavlik, Extracting Comprehensible Concept Representations from Trained Neural Networks, In Working Notes on the IJCAI'95 Workshop on Comprehensibility in Machine Learning (1995) 61-75

[8] Y. Jin, Pareto-Based Multiobjective Machine Learning: An Overview and Case Studies, IEEE transactions on systems, man, and cybernetics - part C: applications and reviews, vol. 38-3 (2008), 397-415.

[9] A. A. Freitas, A critical review of multi-objective optimization in data mining: a position paper, ACM SIGKDD Explorations Newsletter, vol 6-2 (2004), 77-86.

[10] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms,* John Wiley & Sons, Hoboken (2009).

[11] M. Bohanec, I. Bratko, Trading accuracy for simplicity in decision trees, *Machine Learning* vol. 15-3 (1994), 223-250.

[12] R. Piltaver, M. Luštrek, J. Zupančič. S. Džeroski, and M. Gams, Multi-objective learning of hybrid classifiers, *21st European Conference on Artificial Intelligence* (2014).

[13] A. Frank, A. Asuncion, UCI Machine Learning Repository, http://archive.ics.uci.edu/ml

[14] Weka: Collections of Datasets, http://www.cs.waikato.ac.nz/ml/weka/datasets.html

[15] R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[16] I. H. Witten, E. Frank, M. A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, Third Edition. Morgan Kaufmann, San Francisco (2011)

[17] K. Deb, A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, vol. 6- 2 (2002).

[18] J. Yin, *Multi-objective Machine Learning*, Springer, Berlin (2006).

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-230

Predicting Players Behavior in Games with Microtransactions

Ondřej Pluskal [a] Jan Šedivý [a]
[a] *Czech Technical University in Prague, Czech Republic*

Abstract. This paper focuses on predicting player behaviour in two-player games with microtransactions. Typically the games are for free and companies generate their revenue by selling in-game goods. We show creation of a users behaviour model, which are then used in a recommendation system increasing in-game goods purchases. We focus on learning techniques in a novel way, predicting the time of purchases rather than the most likely product to be purchased. The player model is based on in-game signals, such as players success, curiosity, social interactions etc. We had access to a *Pool Live Tour* game dataset made by Geewa. We report promising results in predicting the purchase events.

Keywords. machine learning, feature extraction, online games, data mining

1. Introduction

In this paper we specifically focus on how to improve the monetization of two-player online games. The game is typically free but to gain a special game advantage players may purchase in-game goods improving their skills. They purchase the goods in small payments, microtransactions, for real-currency. To increase the revenue users are bombarded with ads [5]. The problem is that very frequent advertisements are do not typically lead to increased revenue because of advertisement fatigue and advertisement wear out.

Both phenomenons are recognized in marketing models [9]. Advertisement fatigue is the event of a customer no longer liking or buying goods from the advertisement, because the advertisement is bothering them too much. Advertisement wear out means that the customer will ignore the advertisement and it would have no effect.

These two negative effects can be reduced by advertising the in-game goods only when the player is likely to make the purchase. When the ad's timing is correct the likelihood of converting through the advertisement is increased. This improves the revenue. Building such a system by an expert is impossible, simply because we are analysing a large-scale dataset of thousands of players.

We tested our novel idea on a *Pool Live Tour* (PLT) game made by Geewa. PLT is a virtual pool game, that can be played in a web browser or on a tablet. This game is played by 2.5 million daily active users[1] all over the world. The in-game good are better cues giving a slight advantage. Certainly, buying a better cue does not guarantee a win it only increases player's chances. To design the model Geewa provided us with

[1] Collected from http://corporate.geewa.com/game/pool-live-tour/ on 2nd Feb 2014

two datasets, dataset D_1 with a sample of monthly users' activity, totalling 272k unique user ids, and dataset D_2 with monthly activity of a subset of users, that registered that month and bought in-game item, totalling 11.5k unique user ids. The datasets contain client actions collected from the clients as well as events generated by the server. The datasets in their raw form contain over 30 GB.

We show creation of a model from the player logs that predicts the players' readiness to buy a cue after finishing a match. In the next section we review the behaviour prediction in general. We will also discuss impact of some of the recommendation systems. In Section 3 we explain our approach to the behaviour prediction as a machine learning task and develop and evaluate the model. In Section 4 we show the properties of the datasets and how the feature extraction was done. In Section 5 we describe the testing and show the results of our experiments. We conclude our observations and set our goals for future work in Section 6.

2. Related Work

In the literature there are a few studies concerning player modelling using machine learning and data mining methods. We have identified a few trends in the literature, from matchmaking, user segmentation, to cheater detection. Up to our knowledge, nobody published any work about building recommendation systems in games. This section therefore covers not only player modelling, but also recommendation systems.

In cheater detection studies there are several studies using machine learning and data mining methods. The studies define the problem as a classification problem. They use derived rules [7], SVM [13] and Hidden Markov Models [16] with a set threshold. The features were consisting of playtime lengths[7] or action frequencies [13]. Bot detection tasks are studied in several game settings. These are mainly MMOGs[2] [7,13] and FPS[3] games [16].

More descriptive models were made in the case of segmentation [3,4,14], a problem of finding distinct groups of users. Segmentation is viewed in all of these articles as unsupervised machine learning problem. Two papers consider player segmentation task [3,4], the third segments organised groups of players called guilds [14]. Player segmentation is done using cumulative features including playtimes, game specific estimates of success (e.g. number of trials per level, number of kills etc.). The approaches used for this task vary from NMF[14], k-means, Simplex Volume Maximization[3], to SOM[4]. In research the main challenge is the number of users and therefore the problem of large-scale datasets. All the interesting information are extracted from game logs, that the producers store in databases acquired by telemetry. The games used in player segmentation are both single player games [4] as well as multiplayer games [3,14].

There are more papers[10,15] using supervised machine learning. Another paper tries to predict the players' maximal progress in a single player game [10]. This is done using machine learning methods, by monitoring gameplay features in the early levels of the game and solving this problems both as classification problem as well as regression problem. Both problem abstractions show promising results. Another paper [15] studies

[2]Massive Multiplayer Online Games
[3]First Person Shooter

the prediction of strategy that a competitive RTS[4] player would play from his actions done in game at certain time point. This study aims to creating an AI that by predicting the strategy probabilities would infer correct counter strategy.

Since up to our knowledge in the research of player modelling there is no relevant literature of how to present players with in-game goods we have tried to find resources in the setting of web recommendation systems aimed at recommending relevant goods, entertainment or web pages to specific users based on their preferences or browsing history.

In recommendation systems there are many ways from which data we should predict what a user might like. This can be done via contextual information, where in advertisement we try to find ads with similar context to the site. Also user based recommendations, where we look at similar users and recommend goods to them that others purchases (Collaborative filtering). And the last approach is that we use sequential data and try to find sequential patterns that might help us what the user might want to see next. We try to find out similarities with web recommendation, which is in active research for more than ten years[11]. Also hybrid approaches [8] using combinations of different approaches can be seen mainly in the Netflix Prize competition, where researchers improved the accuracy of the Netflix recommendation system by 10%.

Other approaches use sequential patterns [17,12]. The idea comes from web-page recommendation, where a users browsing in some domain might share the same browsing sequential patterns as other users. Finding the most common patterns and creating a patricia trees from them is the core idea. Then assuming the Markov property (dependence of next state only on previous state, here states are web pages) the recommendation system can show the most probable web pages by traversing the patricia tree and suggesting the most probable pages.

To the best of our knowledge there exists no study that would directly solve the issue of dynamic advertisement space generation presented in this paper. This approach can bring the notion of smart recommendation to games, where we can address the player directly in the time, he is most likely to buy a new virtual item.

3. TASK DEFINITION

The main motivation of this paper is to increase the revenue generated by games using microtransactions by reducing the amount of advertisement sent to the user. In the ideal case we would like to suggest to the user to purchase in-game items only in the times he would really buy the in-game item.

We will be explaining our approach on the PLT game. In this game there are two currencies. The first one is coins, that can be acquired by playing, daily rewards and earning trophies. The second one is gold coins, that can be only acquired by purchase using real currency. The task at hand is predicting the buy of a cue for gold coins. We divided the task into two stages, first we try to predict that the player would buy a cue of any kind, then we decided to only predict the gold cue buys. Both stages use only users of which we have information from their registration. For the second stage we use a segment of paying users as a subset of all users whose behaviour we would like to model.

[4]Real Time Strategy

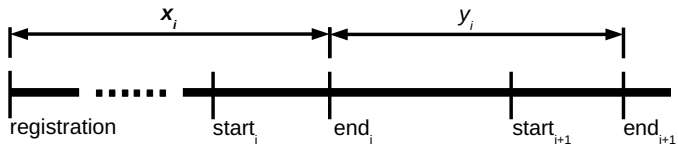


Figure 1. Visualization of $\mathbf{x}_i$ and y_i for player u.

We have decided to model the problem of timely in-game item recommendation as a supervised machine learning task, particularly a binary classification task. The example is a tuple $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ corresponds to a feature vector containing information about the activity of player u from registration to the end of game i. Fig 1 depicts the $\mathbf{x}_i$, y_i relationship on an example of a single user. The y_i is whether the player u bought a cue in between the start end of game i and the end of game $i+1$. Because we have only two targets, we define $y_i = -1$ if the player didn't buy between games i and $i+1$, $y_i = 1$ if the player bought between games i and $i+1$.

Since we described the task as a binary classification tasks, we can use several measures to score classifiers performance. We could use accuracy, precision and recall or area under a ROC curve (AUC) [6]. We decided to use AUC because it generally maximizes the True Positive Rate (TPR) for every False Positive Rate (FPR). This is a good measure for this problem, because upon creating the recommendation system the operator can choose the FPR and then set a threshold on the scoring decision function.

4. DATA COLLECTION

Two datasets of raw data in form of logs were provided by Geewa from their game PLT. The data come from their internal reporting system, that is used by their analytics in order to better understand how the players are playing the game.

These are live data from real players in their homes captured by telemetry and stored on Geewas side. The datasets contain logs from the client residing in player's computers as well as actions generated by the server. The player base is from users from 187 countries all over the world. We were kindly provided with 2 datasets D_1 and D_2 from Geewa. D_1 is drawn as a small sample of all users and all of their actions generated by client and server in one month (19th Mar 2013 - 18th Apr 2013). D_2 is a set of all users that registered and bought some cue in one month (1st Nov 2013 - 30th Nov 2013). The size of raw data is 24 GB for D_1 and 6.6 GB for D_2.

4.1. Data Preprocessing

The log data contain information about user logins, their in-game actions, from starting a game to shooting with a cue. The dataset was provided in a single csv file. We had to split the single file into a file for each player and those needed to be sorted according to the time when they were generated. Both datasets are exported from relational databases in form of one large table with all events in a structured form.

The dataset D_1 consists of a subset of all users. This means that out of 272k users for example we have 181k users who played one game and 78k users who played 10 games. Since we need full user history, we need the users who have newly registered, which out

of the total 272k users is only 57k users who registered. Only 6728 players have played at least 10 games, newly registered and bought some cue. These players were used for the first task of prediction of cue buys.

The dataset D_2 consists of a subset of players who bought a gold cue. This process unfortunately yielded only 6,838 players who registered and played at least 10 games.

4.2. Feature Extraction

We will now describe how we created the feature vector $\mathbf{x}_i$. We created a set of features extracted from the logs containing information about the players progress, success and curiosity. Also there is a difference in over which period the features are extracted. They can be extracted from the registration until the game end, but they can also be extracted from several games before the game end or even in time periods, e.g. in the last 5 hours.

We define a set of all basic features F. These are the basic features extracted from the logs, that are in F:

- **Number of matches:** Number of matches m_a the player played in both main game modes, matched and friend games.
- **Distribution of match types:** Number of each game type the player played. There are 2 different match types in PLT. These are matched games m_m (the player is matched against an opponent according to his skill), friend games m_f(player challenges his friend through some social platform).
- **Distribution of levels played:** A player can play matched games in different game levels in the play mode. There are levels present at the moment. The extracted feature $m_i \; \forall i \in \{1,2,...,14\}$ is the number of games played at each level.
- **Performance in matches:** The number of wins w in different game types. From the game point of view each of the two competitive game types (matched games w_m and friend games w_f) the number of wins for both match types w_a. Also the number of wins per level in matched games is measured w_i. In matched games the player bets a fixed amount of coins and wins his and his opponent's bet, or loses everything. The stakes are the higher the higher is the level.
- **Match properties:** Number of shots m_{ns}, time spent in game t_m.
- **Player curiosity:** Number of shown opponents cards c_{pc}, cue galleries c_{cg}, owners card c_{oc}, shop c_s.
- **Number of trophies:** Number of trophies a a player got through achievements (e. g. winning several games in row, sinking several balls in a row etc.). The player also gets a small reward of additional coins for each trophy.
- **Bonus coins:** A player can get coins for free in two ways, by receiving a daily-bonus package c_{db} or by depleting all coins and receiving a free-bonus package c_{fb}.
- **Amount of coins:** Current coin balance c_b. The coins are used to bet and play games. The bet value is set at each level and if the player wins, he gets back both bets of both players. If the player loses he gains nothing. The player can spend coins for various in-game items, for example cues. Also the player can buy coins for real currency.
- **Amount of winnings:** The cumulative amount of gold a player has won is used to unlock levels. For each level there is a set threshold for winnings, if the player

exceeds the threshold he has the option to play at that level. The feature is denoted as c_w.

- **Rank-ups:** When a player is able to play at a higher level (he has just unlocked the level or he has enough coins to play at a higher level) he is shown a dialog, that he is able to play at higher level. The number of these shown dialogs is encoded in feature c_r.
- **Time from registration:** How much time m_t passed between the end of game and registration.
- **Inventory:** Number of already bought cues b_c and the subset of gold cues b_{gc}.

As we said the feature extraction can be done in three different ways. We will be explaining them using a generic feature f. One is by extracting the cumulative features from registration up until game i, $f(i)$. The second one is by extracting the features over the last k games, $f^{(k)}(i)$. We call this extraction method match windowing. The third is extracting features over a certain time period t, $\hat{f}^{(t)}$. We call this time windowing.

The feature vector is described in equation 1. The t time in time window features $\hat{h}_i^t$ is in hours.

$$\begin{aligned} x = & (f_1,...,f_n,f_1^{(1)},...,f_m^{(1)},f_1^{(10)},...,f_m^{(10)}, \\ & \hat{h}_1^{(1)},...,\hat{h}_l^{(1)},\hat{h}_1^{(24)},...,\hat{h}_1^{(24)},\hat{h}_1^{(168)},...,\hat{h}_l^{(168)}) \\ f_i \in & F \cup R \\ h_i \in & \{m_a,m_m,m_f,m_1,...,m_{14}, \\ & w_a,w_m,w_f,w_1,...,w_{14}\} \cup R \end{aligned} \tag{1}$$

We are using a smaller feature vector for the games played before the first 10 games, since for computing the features $g_i^{(10)}$ we need to have at least 10 games played by the user. We divide each of the datasets D to two parts, $D^{(1-9)}$ and $D^{(10)}$, where feature vectors from $D^{(1-9)}$ do not have the $f_i^{(10)}$ features, but the rest is the same as described in equation 1.

5. EXPERIMENTS

The methodology used for evaluation is based on measuring the AUC. In order to create a valid training protocol, we divided the dataset into two parts, training set and testing set, where training set is 20% and testing set is 80%. If parameter tuning is needed in order to find the best model we use 5-fold cross-validation.

5.1. User based and game based dataset divisions

The most important issue when constructing a testing protocol is the proper division into training, validation and testing sets, so that the examples are i.i.d.(independent and identically distributed). In this task we can split either based on user, or based on games. In the previous section we described how to extract the tuples $D = (\mathbf{x}_i^u, y_i^u), \forall u, \forall i$. The split based on games would be done, by considering every tuple $(\mathbf{x}_i^u, y_i^u)$ independent on u. This could lead to having a feature vector from particular user u in both training and testing set, i.e. game 11 in training and game 12 in testing set. But this contradicts with

Table 1. This table shows the difference between validation and testing error with different splits. It ilustrates overfitting with different techniques.

	Validation AUC	Testing AUC	Change
Game split	74.76%	73.06%	-1.7
User split	75.07%	75.99%	+0.92

Table 2. This table consists of the properties of each of the different datasets.

	D_1^{1-9}	D_1^{10}	D_2^{1-9}	D_2^{10}
No. buys	6,838	11,601	445	3750
No. games	83,160	383,703	35,271	445,554
No. players	9,240	6,161	3,919	3,511

the independency of the two sets. Also we suggest that the performance on new users would be lower for the game split.

We made an experiment to support our claims and the results are shown in table 1 using a Random Forest on $D_1^{(10)}$. The dataset was divided into a training set and testing set using a user split. The validation error is the error using 5 fold cross-validation using the respective splitting on training set. The testing error is the performance of the classifier trained on the whole training set evaluated on the testing set. We can see that our assumption were confirmed empirically.

5.2. General cue and gold cue predictions

Since we are not capable of computing the window features for the early matches, we decided to divide the dataset into two parts. One are games 1-9 and the second part are the games 10 up to what the player accomplished to play in given month. The dataset will be labelled $D^{(1-9)}$ and $D^{(10)}$.For dataset D_1 we will perform the experiment for all cues, since there are only 299 gold buys. For dataset D_2 we will be making experiments on gold cues. The feature vectors will be the same for both settings.

From the characteristics of the dataset shown in table 2, we can see that the datasets have a property of being unbalanced. This means that we have to choose the classifiers accordingly. We have chosen scikit[5] library in python for the training and evaluation of the classifiers.

As the classifier to test the performance we have chosen Random Forest [1], linear SVM [2] and a Decision Tree. They were chosen, because they have the option to handle unbalanced classification tasks, due to the ability to assign class weights. The class weights used were inversely proportional to class frequencies. The Random Forest is a well performing classifier in various different tasks and linear SVM represents a simple but well grounded model. For the classification task we used a small decision tree in order to create a simplified view of the problem. The Decision Tree has maximum depth set to 3 in order to give the reader an idea what are the most discriminative features and how do they work.

Z-normalization was performed on the respective training sets when training the SVM, because the scales in each feature are different. The scales do not affect the tree

[5]Version 0.14 http://scikit-learn.org/stable/

Table 3. Results of the experiments for each classifier and dataset.

	Random Forest	SVM	Decision Tree
$D_1^{(1-9)}$	75.37%	71.22%	71.53%
$D_1^{(10)}$	75.99%	73.25%	68%
$D_2^{(1-9)}$	79.69%	78.65%	76.96%
$D_2^{(10)}$	87.22%	76.72%	78.27%

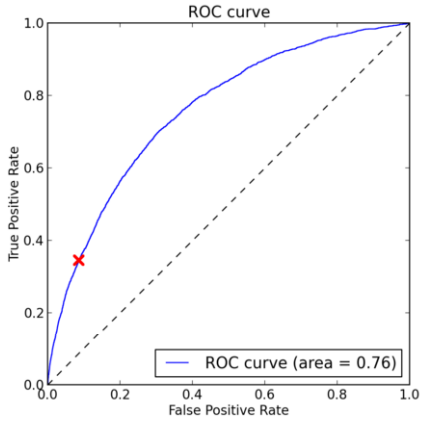


Figure 2. ROC curve of $D_1^{(10)}$

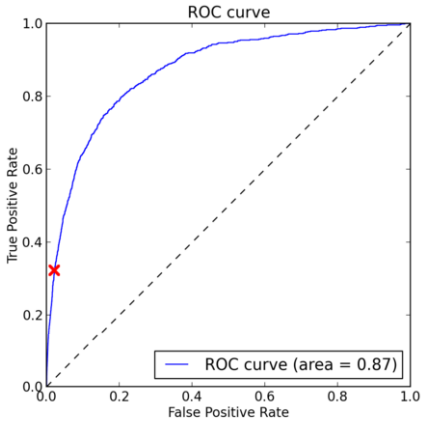


Figure 3. ROC curve of $D_2^{(10)}$

based classifiers. SVM's regularization constant C was tuned on the validation sets. The parameters tuned on the Random Forest were maximum number of feature, minimum number of features and number of estimators.

In table 3 we can see the performance of each classifier on each of the two datasets. We can see that the performance on the first dataset is significantly lower than the performance on the second dataset. We suppose the reason for higher performance of Random Forest over SVM is the fact, that the dataset is clearly not linearly separable. The Random Forest can infer much more complex models than linear ones [1]. Also we can see that the models' performance using only a small decision tree is significantly lower that each of the Random Forests.

In Figures 2 and 3 we can see the ROC curves for all the datasets computed over the testing set. In each figure the cross is the operating point of the classifier.

Both ROC curves show, that the classifiers default operating point was trained on a very low FPR. This corresponds to not suggesting the cues to the player many times. An operator can adjust the threshold to move the TPR as well as FPR higher, in order to raise the advertisement ratio.

The relative feature importance can be computed from each of the trained random forests[1]. We are plotting only the first 20 highest scoring features in order to see, what features are most important and to check whether all the types of designed features are used by the random forest. In figures 4 and 5 we can see, that all types of different features are present. These include to cumulative, time window and game window features.

For both cases $D_1^{(10)}$ and $D_2^{(10)}$ we can see similar features scoring high. One exception is feature importance of the number of previously purchased cues b_{gc} is 0.19 in

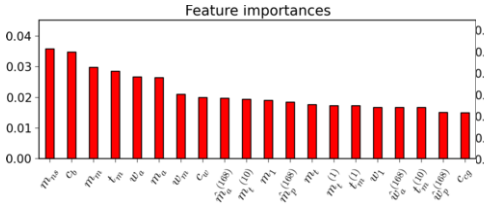


Figure 4. Feature importances of $D_1^{(10)}$

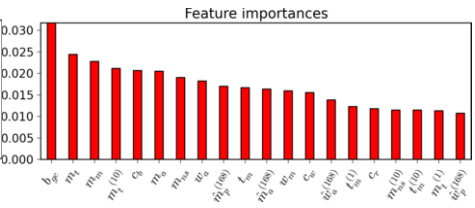


Figure 5. Feature importances of $D_2^{(10)}$

figure 5. This is due to the fact, that many players buy only one cue and the acquisition of a second cue is for the most cases unlikely. Another positive result is the appearance of many different types of features, including cumulative, time window and game window features.

Also we can see that among the highest scoring features there is no feature considering friend matches. The reason is that friend matches are less common than matched matches. In the general cue case we can see that an important feature is the number of clicks on opponent cue gallery. From opponent's cue gallery he can see he is playing against a player with better cue. Using these feature importance plots also might indicate to the developers key components of the game that lead to possible design changes.

6. CONCLUSION AND FUTURE WORK

We presented a system predicting player purchase applicable on games with microtransactions. Unlike a typical recommendation system we are not solving the problem of what we should recommend to the player, but when is the right time to recommend purchase of in-game goods. This approach allows dynamic advertisement placements.

The task is defined as a classification problem deciding whether to display or not an advertisement at the end of a game. This decision is based on the prediction whether a player would buy an in-game item after a match. The features used are of three types, cumulative, game windows and time windows.

We created models predicting two actions general in-game items purchases and in-game items purchases for hard currency leading to real revenue. Our experiments have shown that the Random Forest algorithm was performing the best for both cases. For the general case we have achieved 76% AUC and for the hard currency 87% AUC. We have used the decision trees to study features differentiating power. We have found that the most discriminative features are different for each case. Information about feature importances can be useful to the game designers in order to improve the game and where to focus their design goals.

This system is yet to be tested in practice, because the only way how to measure the real impact on revenue is by A/B testing. We optimistically hope that the correct advertisement placement timing will make the players buy the in-game items for hard currency more frequently or earlier than today. This will bring higher revenue to the company.

We plan to use these models in other tasks such as cheater, bot detection and churn prediction in the future. The developed models can help to solve also these big problems.

7. ACKNOWLEDGEMENTS

This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS14/072/OHK3/1T/13.

References

[1] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[2] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, June 1998.

[3] A. Drachen, R. Sifa, C. Bauckhage, and C. Thurau. Guns, swords and data: Clustering of player behavior in computer games in the wild. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 163–170, 2012.

[4] Anders Drachen, Alessandro Canossa, and Georgios N. Yannakakis. Player modeling using self-organization in tomb raider: underworld. In *Proceedings of the 5th international conference on Computational Intelligence and Games*, CIG'09, pages 1–8, Piscataway, NJ, USA, 2009. IEEE Press.

[5] Canossa Alessandro. El-Nasr Magy Seif, Drachen Anders. *Game Analytics: Maximizing the Value of Player Data*. Springer, 2012.

[6] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.

[7] Ah Reum Kang, Jiyoung Woo, Juyong Park, and Huy Kang Kim. Online game bot detection based on party-play log analysis. *Computers & Mathematics with Applications*, 65(9):1384 – 1395, 2013. Advanced Information Security.

[8] Yehuda Koren and Robert M. Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. Springer, 2011.

[9] Moorthy S. Lillien G., Kotler P. *Marketing Models*. Prentice Hall, 1992.

[10] Tobias Mahlmann, Anders Drachen, Julian Togelius, Alessandro Canossa, and Georgios N. Yannakakis. Predicting player behavior in tomb raider: Underworld. In *CIG*, pages 178–185. IEEE, 2010.

[11] R. Suguna and D. Sharmila. An efficient web recommendation system using collaborative filtering and pattern discovery algorithms. *International Journal of Computer Applications*, 70(3):37–44, May 2013. Published by Foundation of Computer Science, New York, USA.

[12] Usha Rani M. Suneetha K. Web page recommendation approach using weighted sequential patterns and markov model. *Global Journal of Computer Science and Technology*, 2012.

[13] Ruck Thawonmas, Yoshitaka Kashifuji, and Kuan-Ta Chen. Detection of mmorpg bots based on behavior analysis. In *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, ACE '08, pages 91–94, New York, NY, USA, 2008. ACM.

[14] C. Thurau and C. Bauckhage. Analyzing the evolution of social groups in world of warcraft. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 170–177, 2010.

[15] Ben G. Weber and Michael Mateas. A data mining approach to strategy prediction. In *Proceedings of the 5th international conference on Computational Intelligence and Games*, CIG'09, pages 140–147, Piscataway, NJ, USA, 2009. IEEE Press.

[16] S.F. Yeung, J.C.-S. Lui, Jiangchuan Liu, and J. Yan. Detecting cheaters for multiplayer games: theory, design and implementation[1]. In *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE*, volume 2, pages 1178–1182, 2006.

[17] Baoyao Zhou, Siu Cheung Hui, and Kuiyu Chang. An intelligent recommender system using sequential web access patterns. In *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, volume 1, pages 393–398 vol.1, 2004.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-240

Extension–Based Semantics of Abstract Dialectical Frameworks

Sylwia POLBERG [a,1]
[a] *Vienna University of Technology, Institute of Information Systems, Favoritenstraße 9-11, 1040 Vienna, Austria.*

Abstract. One of the most prominent tools for abstract argumentation is the Dung's framework, AF for short. Although powerful, AFs have their shortcomings, which led to development of numerous enrichments. Among the most general ones are the abstract dialectical frameworks, also known as the ADFs. They make use of the so–called acceptance conditions to represent arbitrary relations. This level of abstraction brings not only new challenges, but also requires addressing existing problems in the field. One of the most controversial issues, recognized not only in argumentation, concerns the support or positive dependency cycles. In this paper we introduce a new method to ensure acyclicity of arguments and present a family of extension–based semantics built on it, along with their classification w.r.t. cycles. Finally, we provide ADF versions of the properties known from the Dung setting.

Keywords. abstract argumentation, abstract dialectical frameworks, argumentation semantics

Introduction

Over the last years, argumentation has become an influential subfield of artificial intelligence [2]. One of its subareas is the *abstract argumentation*, which became especially popular thanks to the research of Phan Minh Dung [3]. Although the framework he has developed is quite powerful, it has certain shortcomings, which inspired a search for more general models [4]. Among the most abstract enrichments are the abstract dialectical frameworks, ADFs for short [5]. However, a framework cannot be considered a suitable argumentation tool without properly developed semantics.

The semantics of a framework are meant to capture what we consider rational. Many of the advanced ones, such as grounded or complete, coincide when faced with simple, tree–like frameworks. The differences between them become more visible in complicated cases. On various occasions examples were found for which none of the available semantics returned satisfactory answers. They gave rise to new concepts, such as prudent and careful semantics for handling indirect attacks [6,7], sustainable and tolerant for self–attackers [8] or some of the SCC–recursive semantics for the problem of attack cycles [9]. Introducing a new relation, such as support, creates additional problems.

The most controversial issue in a setting permitting support concerns the support cycles and is handled differently from formalism to formalism. Among the best known

[1]The author is funded by the Vienna PhD School of Informatics. This research is a part of the project I1102 supported by the Austrian Science Fund FWF. An earlier version of this paper can be found in [1].

ones are the Bipolar Argumentation Frameworks (BAFs) [10], Argumentation Frameworks with Necessities (AFNs) [11] and Evidential Argumentation Systems (EASs) [12]. While the latter two discard support cycles, BAFs do not make such restrictions and in general, neither do ADFs. This variety is not an error in any of the structures. First of all, in a more advanced setting, a standard Dung semantics can be extended in several ways. Moreover, since one can find arguments both for and against any of the cycle treatments, lack of consensus as to what approach is the best should not be surprising.

Many properties of the available semantics can be seen as "inside" ones, i.e. "what can I consider rational?". On the other hand, some can be understood as on the "outside", e.g. "what can be considered a valid attacker, what should I defend from?". Various examples of such behavior exist even in the Dung setting. An admissible extension defends against all possible attacks in the framework. We can then restrict this by saying that self–attackers are not rational, and thus limit the set of arguments we have to defend the extension from. If we now add support, we can again define admissibility in the basic manner. However, one often demands that the extensions are free from support cycles and that we only defend from arguments not taking part in them. From this perspective semantics can be seen as a two–person discussion, describing what "I can claim" and "what my opponent can claim". This is also the point of view that we follow in this paper.

Although various extension–based semantics for ADFs have already been proposed in the original paper [5], many of them were defined only for a particular ADF subclass and were not suitable for all types of situations. Moreover, they did not solve the problem of positive dependency cycles. The aim of this paper is to address these issues. We introduce a family of extension–based semantics and specialize them to handle the problem of support cycles, as their treatment seems to be the greatest difference between the available frameworks. Furthermore, we present a classification of our sub–semantics in the internal–external fashion that we have described before. We also recall our research on admissibility in [13] and show how it fits into the new system. Finally, we show which known properties, such as Fundamental Lemma, carry over from the Dung framework.

1. Dung's Argumentation Frameworks

Let us briefly recall the argumentation framework by Dung [3] and its semantics. For more details we refer the reader to [14].

Definition 1.1. A **Dung's abstract argumentation framework** (AF for short) is a pair (A,R), where A is a set of **arguments** and $R \subseteq A \times A$ represents an **attack** relation.

Definition 1.2. An argument $a \in A$ is **defended** by a set E in AF, if for each $b \in A$ s.t. $(b,a) \in R$, $\exists c \in E$ s.t. $(c,b) \in R$. A set $E \subseteq A$ is:

- **conflict–free** in AF iff for each $a,b \in E$, $(a,b) \notin R$.
- **admissible** iff it is conflict–free and defends all of its members.
- **preferred** iff it is maximal w.r.t set inclusion admissible .
- **complete** iff it is admissible and all arguments defended by E are in E.
- **stable** iff it is conflict–free and for each $a \in A \setminus E$ there exists $b \in E$ s.t. $(b,a) \in R$.

The **characteristic function** $F_{AF} : 2^A \to 2^A$ is defined as: $F_{AF}(E) = \{a \mid a \text{ is defended by } E \text{ in } AF\}$. The **grounded extension** is the least fixed point of F_{AF}.

We would also like to recall the notion of range, as its idea will be used in the ADF semantics. Even in the Dung setting, the concepts of the E^+ and E^- sets can be used to redefine defense. Finally, we also list some of the properties of Dung's semantics [3].

Definition 1.3. Let E^+ be the set of arguments attacked by E and E^- the set of arguments that attack E . $E^+ \cup E$ is the **range** of E.

Lemma 1.4. ***Dung's Fundamental Lemma*** *Let E be an admissible extension, a and b two arguments defended by E. Then $E' = E \cup \{a\}$ is admissible and b is defended by E'.*

Theorem 1.5. *Every stable extension is a preferred extension, but not vice versa. Every preferred extension is a complete extension, but not vice versa. The grounded extension is the least w.r.t. set inclusion complete extension. The complete extensions form a complete semilattice w.r.t. set inclusion.* [2]

2. Abstract Dialectical Frameworks

Abstract dialectical frameworks have been defined in [5] and further studied in [13,15, 16,17,18]. Their main goal is to be able to express arbitrary relations. This is achieved by the use of acceptance conditions, which define what sets of arguments should be present in order to accept or reject a given argument.

Definition 2.1. An **abstract dialectical framework** (ADF) as a tuple (S, L, C), where S is a set of abstract **arguments** (nodes, statements), $L \subseteq S \times S$ is a set of **links** (edges) and $C = \{C_s\}_{s \in S}$ is a set of **acceptance conditions**, one condition per each argument.

An acceptance condition is a total function $C_s : 2^{par(s)} \to \{in, out\}$, where $par(s) = \{p \in S \mid (p, s) \in L\}$ is the set of **parents** of an argument s.

Please note that one can also represent the acceptance conditions by propositional formulas over arguments instead of "boolean" functions [19]. It is easy to see that links L are somewhat redundant and can be extracted from the conditions. Thus, we will use of shortened notation and assume an ADF $D = (S, C)$ through the rest of this paper. In order to introduce our new semantics, we need to explain some basic notions first.

Decisiveness: At the heart of Dung's semantics is the concept of defense. Admissibility represents a defensible stand, where no matter what our opponent says against us, we can provide a counter argument to it. From this also follows that given accepted arguments E and the ones attacked by them (E^+), whatever argument is left cannot further change the status of the ones in E. This idea that an argument has a "final" assignment is captured with the notion of decisiveness in ADFs. Its basic form, where we check the outcome of an acceptance condition w.r.t. accepted and rejected sets of arguments, can already be found in the original paper [5]. It is further developed in an interpretation–based form in [13], which will be used in this paper. A two–valued interpretation v is a mapping that assigns the truth values $\{\mathbf{t}, \mathbf{f}\}$ to arguments. We will use v^x to denote a set of arguments mapped to x by v, where x is some truth–value. Given an argument $s \in S$, its condition C_s and an interpretation v, we define a shorthand $v(C_s)$ as $C_s(v^{\mathbf{t}} \cap par(s))$. Now, in order

[2] A partial order $(A, \leq)$ is a complete semilattice iff each nonempty subset of A has a glb and each increasing sequence of S has a lub.

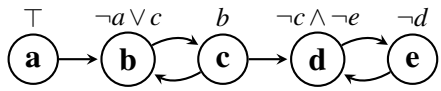


Figure 1. Sample ADF

to check is some (possibly partial) interpretation is decisive for s, we basically need to check if all "bigger" interpretations stemming from it evaluate C_s in the same way:

Definition 2.2. Given a two–valued interpretation v defined on a set A, a **completion** of v to a set Z where $A \subseteq Z$ is an interpretation v' defined on Z s.t. $\forall a \in A\ v(a) = v'(a)$. By a $\mathbf{t}/\mathbf{f}$ completion we understand v' that maps all arguments in $Z \setminus A$ respectively to $\mathbf{t}/\mathbf{f}$.

Definition 2.3. We say that an interpretation v defined on A is **decisive** for an argument $s \in S$ iff for any two (respectively two or three–valued) completions $v_{par(s)}$ and $v'_{par(s)}$ of v to $A \cup par(s)$, it holds that $v_{par(s)}(C_s) = v'_{par(s)}(C_s)$. We say that s is **decisively out/in** wrt v if v is decisive and all of its completions evaluate C_s to respectively out, in.

Example 2.4. Let $(\{a,b,c,d,e\}, \{C_a : \top, C_b : \neg a \vee c, C_c : b, C_d : \neg c \wedge \neg e, C_e : \neg d\})$ be the ADF in Figure 1. Examples of decisively in interpretations for b include $v_1 = \{c : \mathbf{t}\}$. This means that knowing that c is true, we know that the whole disjunction (and thus the acceptance condition) are satisfied. Formally speaking, v_1 is decisive as both of its completions $\{c : \mathbf{t}, a : \mathbf{f}\}$ and $\{c : \mathbf{t}, a : \mathbf{t}\}$ satisfy the condition.

Acyclicity: Let us now focus on the issue of positive dependency cycles. Please note we refrain from calling them support cycles in the ADF setting in order not to confuse them with specific definitions of support available in the literature [10].

Informally speaking, an argument takes part in a cycle if its acceptance depends on itself. An intuitive way of verifying the acyclicity of an argument would be to "track" its evaluation, e.g. in order to accept a we need to accept b, for b we need c and so on. This basic case becomes more complicated when disjunction is introduced. We then receive a number of such "paths", with only some of them proving to be acyclic. Moreover, they might be conflicting one with each other. It can also happen that all acyclic evaluations are blocked and a cycle is forced. Our approach to acyclicity is based on the idea of such "paths" that are accompanied by sets of arguments used to detect possible conflicts.

Let us now introduce the formal definitions. In order to obtain the arguments that are required or should be avoided for the acceptance of a given argument, we will make use of decisive interpretations. Naturally, it suffices to focus on the minimal ones, by which we understand that both $v^{\mathbf{t}}$ and $v^{\mathbf{f}}$ are minimal w.r.t. $\subseteq$. Given an argument $s \in S$ and $x \in \{in, out\}$, by $min\_dec(x,s)$ we will denote the set of minimal two–valued interpretations that are decisively x for s.

Definition 2.5. Let $A \subseteq S$ be a nonempty set of arguments. A **positive dependency function** on A is a function pd assigning every argument $a \in A$ an interpretation $v \in min\_dec(in,a)$ s.t. $v^t \subseteq A$ or $\mathscr{N}$ for null iff no such interpretation can be found.

Definition 2.6. An **acyclic positive dependency evaluation** ace^a for $a \in A$ based on a given pd–function pd is a pair $((a_0,...,a_n),B)$, [3] where $B = \bigcup_{i=0}^{n} pd(a_i)^{\mathbf{f}}$ and $(a_0,...,a_n)$

[3] Please note that it is not required that $B \subseteq A$

is a sequence of distinct elements of A s.t.: *1)* $\forall_{i=0}^{n} pd(a_i) \neq \mathscr{N}$, *2)* $a_n = a$, *3)* $pd(a_0)^{\mathbf{t}} = \emptyset$, and *4)* $\forall_{i=1}^{n}$, $pd(a_i)^{\mathbf{t}} \subseteq \{a_0, ..., a_{i-1}\}$. We will refer to the sequence as the **pd–sequence** and to B as the **blocking set**. We will say that an argument a is **pd–acyclic** in A iff there exist a pd–function on A and a corresponding acyclic pd–evaluation for a.

We will write that an argument has an acyclic pd–evaluation on A if there is some pd–function on A from which we can produce the evaluation. There are two ways we can "attack" an acyclic evaluation. We can either discard an argument required by the evaluation or accept one that is capable of preventing it. This corresponds to rejecting a member of a pd–sequence or accepting an argument from the blocking set. We can now formulate this "conflict" by the means of an interpretation:

Definition 2.7. Let $A \subseteq S$ be some set of arguments and $a \in A$ s.t. a has an acyclic pd–evaluation $ace^a = ((a_0, ..., a_n), B)$ in A. We say that a two–valued interpretation v **blocks** ace^a iff $\exists b \in B$ s.t. $v(b) = \mathbf{t}$ or $\exists a_i \in \{a_0, ..., a_n\}$ s.t. $v(a_i) = \mathbf{f}$.

Example 2.4 (Continued)**.** Let us now show why we store the blocking set. For argument b there exist two minimal decisively in interpretations: $v_1 = \{a : \mathbf{f}\}$ and $v_2 = \{c : \mathbf{t}\}$. The interpretations for a and c are respectively $w_1 = \{\}$ and $z_1 = \{b : \mathbf{t}\}$. Therefore, on $\{a, b, c\}$ we have two pd–functions, namely $pd_1 = \{a : w_1, b : v_1, c : z_1\}$ and $pd_2 = \{a : w_1, b : v_2, c : z_1\}$. They result in one acyclic evaluation for a: $((a), \emptyset)$, one for b: $((b), \{a\})$ and one for c: $((b, c), \{a\})$. Let us analyze the set $E = \{a, b, c\}$. We can see that accepting a "forces" a cycle between b and c. The acceptance conditions of all arguments are satisfied, thus this simple check is not enough to verify if a cycle occurs. If we checked only if the members of the pd–sequences are accepted, we would also get the wrong answer. Only looking at the whole evaluations shows us that b and c are both blocked by a. Although b and c are technically pd–acyclic in E, we see that their evaluations are in fact blocked and this type of conflict needs to be taken into account by the semantics.

3. Extension–Based Semantics of ADFs

Although various semantics for ADFs have already been defined in the original paper [5], only three of them – conflict–free, model and grounded – are still used (issues with the other notions can be found in [13,15,16]). Moreover, the treatment of cycles and their handling by the semantics was not sufficiently developed. In this section we will address all of those issues. Recall that there is no consensus among available bipolar frameworks as to how support cycles should be treated. Therefore, we would like to explore the possible approaches in the context of ADFs by developing appropriate semantics.

The classification of the sub–semantics that we will adopt in this paper is based on the inside–outside intuition presented in the introduction. Appropriate semantics will receive a two–element prefix $xy-$, where x will denote whether cycles are permitted or not on the "inside" and y on the "outside". We will use $x, y \in \{a, c\}$, where a will stand for *acyclic* and c for *cyclic* constraints. As the conflict–free and naive semantics focus only on what we can accept, we will drop the prefixing in this case. Although the model, stable and grounded fit into our classification (see Theorem 3.15 and [20]), they have quite unique naming and further annotations are not needed. We are thus left with admissible, preferred and complete. The BAF approach follows the idea that we

can accept arguments that are not acyclic and we allow our opponent to do the same. The ADF semantics created in [13] also shares this view. Therefore, they will receive the $cc-$ prefix. In contrast, AFNs and EASs do not permit cycles both in extensions and in attackers. Thus, the semantics following this line of reasoning will be prefixed with $aa-$[4]. Although we believe that a non–uniform approach can be suitable in certain situations (i.e. $ca-$ and $ac-$), for now we will focus only on the $aa-$ and $cc-$ ones.

Conflict–free and naive semantics: In the Dung setting, conflict–freeness meant that the elements of an extension could not attack one another. In ADF setting, this notion is strengthened by also providing required support. This represents the intuition of arguments that can stand together presented in [14].

Definition 3.1. A set of arguments $E \subseteq S$ is a **conflict–free extension** of D iff for all $s \in E$ we have $C_s(E \cap par(s)) = in$.

In the acyclic version of conflict–freeness we also need to deal with the conflicts arising on the level of evaluations. To meet the formal requirements, we first have to show how the notions of range and the E^+ set are moved to ADFs.

Definition 3.2. Let $E \subseteq S$ a conflict–free extension of D and v_E a partial two–valued interpretation built as follows:

1. Let $M = E$ and for every $a \in M$ set $v_E(a) = \mathbf{t}$;
2. For every $b \in S \setminus M$ that is decisively out in v_E, set $v_E(b) = \mathbf{f}$ and add b to M;
3. Repeat Step 2 until no new elements are added to M.

By E^+ we understand the set of arguments $v_E^{\mathbf{f}}$ and we will refer to it as the **discarded set**. v_E now forms the **range interpretation** of E.

However, this notion of range is quite strict as it requires an explicit "attack" on all possible arguments. This is not always a desirable property, since depending on the approach we might not treat cyclic arguments as valid and hence want them "out of the way".

Definition 3.3. Let $E \subseteq S$ a conflict–free extension of D and v_E^a a partial two–valued interpretation built as follows:

1. Let $M = E$. For every $a \in M$ set $v_E^a(a) = \mathbf{t}$.
2. For every $b \in S \setminus M$ s.t. every acyclic pd–evaluation of b in S is blocked by v_E^a, set $v_E^a(b) = \mathbf{f}$ and add b to M.
3. Repeat Step 2 until no new elements are added to M.

By E^{a+} we understand the set $(v_E^a)^{\mathbf{f}}$ and call it the **acyclic discarded set**. v_E^a now forms the **acyclic range interpretation** of E.

We can now define an acyclic version of conflict–freeness and the naive semantics:

Definition 3.4. A conflict–free extension E is a **pd–acyclic conflict–free extension** of D iff every argument $a \in E$ has an unblocked acyclic pd–evaluation on E w.r.t. v^E.[5]

[4]More explanations can be found in [20].

[5]Since we are in a conflict–free setting, it suffices to check whether $E \cap B = \emptyset$ to see if an evaluation is not blocked. Consequently, it does not matter which version of range we use.

Definition 3.5. The **naive** and **pd–acyclic naive** extensions are respectively maximal w.r.t. set inclusion conflict–free and pd–acyclic conflict–free extensions.

Example 2.4 (Continued)**.** The conflict–free extensions of our ADF $(\{a,b,c,d,e\},\{C_a : \top, C_b : \neg a \vee c, C_c : b, C_d : \neg c \wedge \neg e, C_e : \neg d\})$ are $\emptyset, \{a\}, \{b\}, \{d\}, \{e\}, \{a,d\}, \{a,e\}, \{b,c\}, \{b,d\}, \{b,e\}, \{a,b,c\}, \{b,c,e\}$ and $\{a,b,c,e\}$. As a blocks evaluations of b and c, $\{a,b,c\}$ and $\{a,b,c,e\}$ are not pd–acyclic conflict–free. Naive and pd–acyclic naive extensions are respectively $\{\{a,d\},\{b,d\},\{a,b,c,e\}\}$ and $\{\{a,d\},\{a,e\},\{b,d\},\{b,c,e\}\}$.

Let us briefly look at the discarded sets of $\{a,d\}$. Since a blocks the evaluations of b and c, they will be included in the acyclic version. However, since none of them is decisively out w.r.t. just $\{a\}$, they will not appear in the standard version. It is easy to see that presence of d permanently discards e and thus it is in both sets.

Model and stable semantics: The concept of a model basically follows the intuition that if something can be accepted, it should be accepted:

Definition 3.6. A conflict–free extension E is a **model** of D if $\forall\, s \in S,\ C_s(E \cap par(s)) = in$ implies $s \in E$.

Verifying whether a condition of an argument s is met does check the effect of accepting s on E, thus it can happen that including s breaks conflict–freeness of E. Consequently, it is clear to see that model semantics is not universally defined. Moreover, the extensions might not be maximal w.r.t. $\subseteq$, as visible in the continuation of Example 2.4.

The model semantics was used as a mean to obtain the stable models. The main idea was to make sure that the model is acyclic. Although the original reduction–based method was not adequate [15], the initial idea still holds and we use it to define stability. Although the produced extensions are now incomparable w.r.t. set inclusion, the semantics is still not universally defined.

Definition 3.7. A model E is a **stable extension** iff it is pd–acyclic conflict–free.

Example 2.4 (Continued)**.** Out of all conflict–free extensions, only $\{a,d\}, \{a,e\}$ and $\{a,b,c,e\}$ are models. $\{a\}$ itself is not a model, since $C_d(\{a\} \cap \{c,e\}) = in$ and $C_e(\{a\} \cap \{d\}) = in$ and we thus we can still accept some arguments. Recall that $\{a,b,c,e\}$ was not pd–acyclic conflict–free. Consequently, $\{a,d\}$ and $\{a,e\}$ are our stable extensions.

Grounded semantics: The grounded semantics introduced in [5] is defined in the terms of a special operator. Although it might look complicated at first, this is nothing more than analyzing decisiveness using the set, not the interpretation form [20].

Definition 3.8. Let $\Gamma_D(A,R) = (acc(A,R), reb(A,R))$, where $acc(A,R) = \{r \in S \mid A \subseteq S' \subseteq (S \backslash R) \Rightarrow C_r(S' \cap par(s)) = in\}$ and $reb(A,R) = \{r \in S \mid A \subseteq S' \subseteq (S \backslash R) \Rightarrow C_r(S' \cap par(s)) = out\}$. Then E is the **grounded model** of D iff for some $E' \subseteq S, (E,E')$ is the least fix–point of Γ_D.

Example 2.4 (Continued)**.** As noted in [5], the grounded extension can be obtained by applying the operator to $(\emptyset,\emptyset)$. Let us compute $acc(\emptyset,\emptyset)$. Obviously, we can accept a. The condition of c is already *out*, and so are the ones of b, d and e when we consider S' being respectively $\{a\}, \{e\}$ and $\{d\}$. It is easy to see that for now, $reb(\emptyset,\emptyset)$ remains empty – S' such as $\{a,c\}, \{b\}, \emptyset$ and $\emptyset$ evaluate respectively b, c, d and e to *in*. Next

step is $\Gamma_D(\{a\},\emptyset)$. However, by reasoning presented above no further arguments can be accepted or rejected and we reach a fixpoint. Thus, $\{a\}$ is our grounded extension.

Admissible and preferred semantics: In [13] we have presented our first definition of admissibility, before the sub–semantics classification was developed. The new, simplified version of our previous formulation, is now as follows:

Definition 3.9. A conflict–extension $E \subseteq S$ is **cc–admissible** in D iff every $e \in E$ is decisively in w.r.t to the range interpretation v_E.

It should be noted that decisiveness w.r.t. range encapsulates the defense known from the Dung setting. If an argument is decisively in, then any set of arguments that would have the power to out the acceptance condition is "prevented" by the interpretation. Hence, the statements required for the acceptance of a are mapped to **t** and those that would make us reject a are mapped to **f**. The former encapsulates the required "support", while the latter contains the "attackers" known from the Dung setting.

When working with the acyclic semantics, we not only have to defend the members, but also their acyclic evaluations. Example 2.4 shows that although decisiveness encapsulates defense of an argument, it might not be the case for its evaluation.

Definition 3.10. A pd–acyclic conflict–free extension E is **aa–admissible** in D iff every $e \in E$ *1)* is decisively in w.r.t. acyclic range interpretation v^a_E, and *2)* has an unblocked acyclic pd–evaluation on E s.t. all members of its blocking set B are mapped to **f** by v^a_E.

Definition 3.11. A set of arguments is **xy–preferred** in D iff it is maximal w.r.t. set inclusion xy–admissible, where $x,y \in \{a,c\}$.

Example 2.4 (Continued)**.** Recall our ADF $(\{a,b,c,d,e\},\{C_a:\top, C_b:\neg a\vee c, C_c:b, C_d:\neg c\wedge\neg e, C_e:\neg d\})$ and that $\{b,c\}$ was a pd–acyclic conflict–free extension. Its standard and acyclic discarded sets are just $\{d\}$. It is easy to see, that both arguments are decisively in w.r.t. both range interpretations. Although uttering a would not change the values of the conditions, it would still force a cycle between b and c. Thus, the acyclicity is not "defended" and $\{b,c\}$ is cc, but not aa–admissible. Similar follows for $\{b,c,e\}$. $\emptyset$ and $\{a\}$ are trivially both cc and aa–admissible. Since the discarded sets of $\{e\}$ include d, so are $\{e\}$ and $\{a,e\}$. By the reasoning above, it is also easy to see that $\{a,b,c\},\{b,c,e\}$ and $\{a,b,c,e\}$ are cc (though not aa) admissible. Finally, apart from $\emptyset,\{a\}$ and $\{a,e\}$, also $\{a,d\}$ is aa–admissible. Its acyclic discarded set is $\{b,c,e\}$ and thus decisiveness and evaluation defense are preserved. Since the standard set is just $\{e\}$, d is not decisively in and the extension is not cc–admissible. The set $\{a,b,c,e\}$ is the only cc–preferred extension, while $\{a,d\}$ and $\{a,e\}$ are aa–preferred.

Complete semantics: Completeness represents an approach in which we have to accept everything we can safely conclude from our opinions. In the Dung setting "safety" means defense, while in the bipolar setting it is strengthened by providing sufficient support. In a sense, it follows the model intuition that whatever we can accept, we should accept. However, now we not only use an admissible base in place of a conflict–free one, but also defend the arguments in question. Therefore, instead of checking if an argument is in, we want it to be decisively in.

Definition 3.12. A cc–admissible extension E is **cc–complete** in D iff every argument in S that is decisively in w.r.t. to range interpretation v_E is in E. An aa–admissible extension E is **aa–complete** in D iff every argument in S that is decisively in w.r.t. to acyclic range interpretation v_E^a is in E.[6]

Example 2.4 (Continued)**.** It is easy to see that since a can always be accepted, only $\{a,d\},\{a,e\}$ and $\{a\}$ are aa–complete. From this also follows that cc–admissible extensions such as $\emptyset,\{e\},\{b,c\}\{b,c,e\}$ are not cc–complete. Since the discarded set of $\{a,b,c\}$ is $\{d\}$, e can still be included in the extension and thus it is also disqualified. Therefore, we obtain three cc–complete sets: $\{a\}$, $\{a,e\}$ and $\{a,b,c,e\}$.

Properties: Let us close the paper with the properties of our semantics. Although the study provided here will by no means be exhaustive, we would like to show how the lemmas and theorems from the original paper on AFs [3] are shifted into this new setting[7].

Even though every pd–acyclic conflict–free extension is also conflict–free, it does not mean that every aa–admissible is cc–admissible. These approaches differ significantly. The first one makes additional restrictions on the "inside", but due to acyclicity requirements on the "outside" there are less arguments a given extension has to defend from. The latter allows more freedom as to what we can accept, but also gives this freedom to the opponent, thus there are more possible attackers. Moreover, it should not come as a surprise that these differences pass over to the preferred and complete semantics, as visible in Example 2.4. Our results show that admissible sub–semantics satisfy the Fundamental Lemma. Moreover, the relations between the semantics presented in [3] are preserved by some of the specializations.

Lemma 3.13. *CC Fundamental Lemma: Let E be a cc–admissible extension, v_E its range interpretation and $a,b \in S$ two arguments decisively in w.r.t. v_E. Then $E' = E \cup \{a\}$ is cc–admissible and b is decisively in w.r.t. v'_E.*

Lemma 3.14. *AA Fundamental Lemma: Let E be an aa-admissible extension, v_E^a its acyclic range interpretation and $a,b \in S$ two arguments decisively in w.r.t. v_E^a. Then $E' = E \cup \{a\}$ is aa–admissible and b is decisively in w.r.t. v'_E.*

Theorem 3.15. *Every stable extension is an aa–preferred extension, but not vice versa. Every xy–preferred extension is an xy–complete extension for $x,y \in \{a,c\}$, but not vice versa. The grounded extension might not be an aa–complete extension. The grounded extension is the least w.r.t. set inclusion cc–complete extension.*

4. Conclusions and future work

In this paper we have introduced a method for detecting positive dependency cycles in ADFs and a family of semantics based on it. Our results show that they satisfy ADF versions of Dung's Fundamental Lemma and that appropriate sub–semantics preserve the relations between stable, preferred and complete approaches. Our future work focuses on shifting the mentioned bipolar frameworks into the ADF setting and proving that their

[6] No further assumptions as to the defense of the evaluations are needed, as visible in Lemma 3.14.

[7] The relevant proofs can be found in [20].

semantics are properly generalized by the presented approaches. Moreover, we would like to study the complexity of the new semantics. Final aim is to provide an efficient implementation, as the existing one was created purely for verification purposes and leaves a lot of room for optimization.

References

[1] Sylwia Polberg. Extension-based semantics of abstract dialectical frameworks. In *Proc. of NMR*, 2014.

[2] Iyad Rahwan and Guillermo R. Simari, editors. *Argumentation in Artificial Intelligence*. Springer, 2009.

[3] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77:321–357, 1995.

[4] Gerhard Brewka, Sylwia Polberg, and Stefan Woltran. Generalizations of dung frameworks and their role in formal argumentation. *Intelligent Systems, IEEE*, 29(1):30–38, Jan 2014.

[5] Gerhard Brewka and Stefan Woltran. Abstract dialectical frameworks. In *Proc. KR '10*, pages 102–111. AAAI Press, 2010.

[6] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Inference from controversial arguments. In Geoff Sutcliffe and Andrei Voronkov, editors, *Proc. LPAR '05*, volume 3835 of *LNCS*, pages 606–620. Springer Berlin Heidelberg, 2005.

[7] Sylvie Coste-Marquis, Caroline Devred, and Pierre Marquis. Prudent semantics for argumentation frameworks. In *Proc. of ICTAI'05*, pages 568–572, Washington, DC, USA, 2005. IEEE Computer Society.

[8] Gustavo A. Bodanza and Fernando A. Tohmé. Two approaches to the problems of self-attacking arguments and general odd-length cycles of attack. *Journal of Applied Logic*, 7(4):403 – 420, 2009. Special Issue: Formal Models of Belief Change in Rational Agents.

[9] Pietro Baroni, Massimiliano Giacomin, and Giovanni Guida. SCC-Recursiveness: A general schema for argumentation semantics. *Artif. Intell.*, 168(1-2):162–210, 2005.

[10] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Bipolarity in argumentation graphs: Towards a better understanding. *Int. J. Approx. Reasoning*, 54(7):876–899, 2013.

[11] Farid Nouioua. AFs with necessities: Further semantics and labelling characterization. In Weiru Liu, V.S. Subrahmanian, and Jef Wijsen, editors, *Proc. SUM '13*, volume 8078 of *LNCS*, pages 120–133. Springer Berlin Heidelberg, 2013.

[12] Nir Oren and Timothy J. Norman. Semantics for evidence-based argumentation. In *Proc. COMMA '08*, volume 172 of *Frontiers in Artificial Intelligence and Applications*, pages 276–284. IOS Press, 2008.

[13] Sylwia Polberg, Johannes Peter Wallner, and Stefan Woltran. Admissibility in the abstract dialectical framework. In *Proc. CLIMA'13*, volume 8143 of *LNCS*, pages 102–118. Springer, 2013.

[14] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *Knowledge Eng. Review*, 26(4):365–410, 2011.

[15] Gerhard Brewka, Stefan Ellmauthaler, Hannes Strass, Johannes Peter Wallner, and Stefan Woltran. Abstract dialectical frameworks revisited. In *Proc. IJCAI'13*, pages 803–809. AAAI Press, 2013.

[16] Hannes Strass. Approximating operators and semantics for abstract dialectical frameworks. *Artificial Intelligence*, 205:39 – 70, 2013.

[17] Hannes Strass. Instantiating knowledge bases in abstract dialectical frameworks. In *Proc. CLIMA'13*, volume 8143 of *LNCS*, pages 86–101. Springer, 2013.

[18] Hannes Strass and Johannes Peter Wallner. Analyzing the Computational Complexity of Abstract Dialectical Frameworks via Approximation Fixpoint Theory. In *Proc. KR '14*, Vienna, Austria, July 2014. Forthcoming.

[19] Stefan Ellmauthaler. Abstract dialectical frameworks: properties, complexity, and implementation. Master's thesis, Faculty of Informatics, Institute of Information Systems, Vienna University of Technology, 2012.

[20] Sylwia Polberg. Extension–based semantics of abstract dialectical frameworks. Technical Report DBAI-TR-2014-85, Institute for Information Systems,Vienna University of Technology, 2014.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-250

The Margin of Victory in Schulze, Cup, and Copeland Elections: Complexity of the Regular and Exact Variants

Yannick Reisch Jörg Rothe Lena Schend
Heinrich-Heine-Universität Düsseldorf, Germany
email: yannick.reisch@hhu.de, {rothe,schend}@cs.uni-duesseldorf.de

Abstract. The margin of victory is a critical measure for the robustness of voting systems in terms of changing election outcomes due to errors in the ballots or fraud in using electronic voting machines. Applications include risk-limiting post-election audits so as to restore the trust in the correctness of election outcomes.

Continuing the work of Xia [24], we show that the margin of victory problem is NP-complete for Schulze and cup elections. We also consider the exact variant of this problem, which we show to be complete for DP in Schulze, cup, and Copeland elections.

Keywords. margin of victory, computational complexity, Schulze elections, cup elections, Copeland elections, computational social choice

1. Introduction

The computational aspects of voting, as a common method of preference aggregation, are a central topic in the field of computational social choice, mainly due to the wide range of applications in multiagent systems (see, e.g., the bookchapter by Brandt et al. [1]). Much of the work so far has focused on winner determination and various types of manipulative attacks (including manipulation [4], bribery [5], and control [6]). Other properties of voting systems have been studied extensively from a social-choice perspective, but much less so in terms of their computational complexity.

We are concerned with one such property: the *robustness of elections*. When voters cast their votes using voting machines in political elections, errors might occur in various ways (be it by accident or with malicious intent [23]), leading to incorrect vote counts. How many errors are affordable before the election outcome (i.e., the winner set) changes? The *margin of victory* is a central concept to measure the robustness of voting systems in terms of changing election outcomes due to errors in the ballots, or due to fraud. It is defined as the smallest number of votes that need to be changed in a given election so as to change its winner set. The higher the margin of victory is, the more robust is the election. In political elections, post-election audits are used to restore the trust in the correctness of election outcomes via "verifiable paper records" [10], and if too many mismatches are found, an extremely costly recount of all votes is in order. Risk-limiting audit methods [20,21,15] do not require to recount *all* votes, while limiting

the risk that the result might still be wrong. A critical parameter for this is the margin of victory.

These issues have been intensively studied from the point of view of political sciences, employing mainly statistical methods and focusing on plurality voting [18,19], scoring rules, approval voting, range voting, and single transferable vote (STV) [15,3,8]. Xia [24] was the first to study the margin of victory in terms of its computational complexity while other notions of robustness in voting have been studied by Procaccia et al. [13] and Shiryaev et al. [17] from a computational perspective. In particular, Xia showed that while this problem is efficiently solvable for scoring rules, approval voting, plurality with run-off, and Bucklin voting, it is NP-complete for Copeland, STV, maximin, and ranked pairs, and he also studied the approximability of these NP-complete problems. Continuing this line of research, we study the complexity of the margin of victory for Schulze and cup elections (a.k.a. sequential majority), establishing NP-completeness as well. The Schulze rule is a particularly attractive voting system due to its many desirable axiomatic properties [16]; its computational properties have also drawn much attention recently (see, e.g., [12]).

While our result for cup voting requires a novel reduction, NP-hardness of the margin of victory problem for Schulze is a straightforward consequence of the NP-hardness result for destructive bribery due to Parkes and Xia [12], along with the connection between these two problems due to Xia [24]. We add to this connection by showing that, for multi-winner voting systems, destructive bribery can be easy, yet the margin of victory problem can still be hard.

The main technical contribution of this paper is the study of the *exact* variant of the margin of victory problem for Schulze, cup, and Copeland elections for which we obtain DP-completeness results. Exact variants of NP-hard problems from a variety of areas are known to be DP-complete (often with rather involved proofs; see, e.g., [22] and the survey [14]), including the exact variants of social welfare optimization problems in multiagent resource allocation [9]. In the exact margin of victory problem, we not only ask whether the margin of victory of a given election meets or exceeds some given threshold, but we ask whether or not it falls into a predetermined interval. It is known that the size of this interval does not matter in terms of the problems' complexity (see, e.g., [22]), so we can fine-tune it to whatever accuracy we desire, even to just one integer, and that is how we will define this problem.

2. Preliminaries

Elections and voting systems: An *election* is a pair (C,V), where C is a finite set of candidates and V is a list of votes (or ballots) expressing the voters' preferences over the candidates in C. The form of the ballots depends on the voting system used; we focus on ballots that are (strict) linear orders of the candidates in C. For example, if $C = \{a,b,c\}$ is our candidate set, a ballot could be of the form $b > c > a$ meaning that this voter (strictly) prefers b to c, and (strictly) prefers c to a. Throughout this paper, we will omit the greater-than sign, so the above preference would be written as $b\,c\,a$.

For a given election (C,V) and two candidates $a,b \in C$, let $D_V(a,b)$ denote the number of votes in V that prefer a to b minus the number of votes in V that prefer b to a. If $D_V(a,b) > 0$, we say that a (strictly) beats b in pairwise comparison. Given an election

(C,V), define the *weighted majority graph for* (C,V), denoted by $\mathrm{WMG}(C,V)$, to be the weighted, directed graph G with vertex set C and edges between any two distinct vertices, where the weight of an edge (a,b) is $D_V(a,b)$ and $D_V(a,b) = -D_V(b,a)$ holds by definition.

A *voting system* $\mathscr{E}$ is a (set of) rule(s) for how to determine the winner(s) of an election (C,V) based on the ballots in V. We will denote the *set of winners of* (C,V) *under* $\mathscr{E}$ by $\mathscr{E}(C,V)$. In particular, we will consider the following voting systems.

Let α, $0 \leq \alpha \leq 1$, be a fixed rational number. In *Copeland*$^\alpha$ elections, given an election (C,V), $D_V(a,b)$ is determined for every pair $(a,b) \in C \times C$. Each candidate a receives one point for every pairwise comparison she (strictly) wins (i.e., whenever $D_V(a,b) > 0$), and gets α points for every tie (i.e., whenever $D_V(a,b) = 0$). All candidates with the highest score are the Copeland$^\alpha$ winners of (C,V).

In *Schulze* elections, construct the weighted majority graph $G = \mathrm{WMG}(C,V)$ from a given election (C,V). The *strength of a path from a to b in G* is defined as the smallest weight any edge on this path has. For each pair (a,b) of candidates, $P(a,b)$ denotes the strength of a *strongest* path from a to b (i.e., of a path with the greatest minimum edge weight among all paths from a to b). All candidates $a \in C$ with $P(a,b) \geq P(b,a)$ for all $b \in C \setminus \{a\}$ are the Schulze winners of (C,V). Note that a candidate $a \in C$ is the unique Schulze winner of (C,V) if and only if $P(a,b) > P(b,a)$ for all $b \in C \setminus \{a\}$.

In *cup* (or *sequential majority*) elections, an election is defined by specifying (C,V) and, additionally, a voting tree T (i.e., a complete binary tree with as many leaves as there are candidates in C, where we assume that C contains enough dummy candidates so as to satisfy $\|C\| = 2^k$ for some k, and all dummy candidates are ranked worst in V), and a schedule that assigns the candidates to the leaves of T. Determine the value of $D_V(a,b)$ for each pair of candidates, a and b, that are siblings in the tree. The winner of the pairwise comparison is assigned to the parent node. This procedure is continued until the cup winner is assigned to the root. The schedule is known beforehand and whenever ties occur, they are broken by a beforehand fixed tie-breaking rule.

Complexity theory: We assume that the reader is familiar with the basic notions of the complexity classes P and NP and hardness and completeness with respect to the polynomial-time many-one reduction, denoted by $\leq^{\mathrm{p}}_{\mathrm{m}}$. Papadimitriou and Yannakakis [11] introduced the complexity class $\mathrm{DP} = \{A \setminus B \,|\, A, B \in \mathrm{NP}\}$, the class of differences of any two NP problems, which is, together with coDP, also known as the second level of the boolean hierarchy over NP (see [2]). It is well-known (see, e.g., [14]) that DP contains the exact variants of many NP-complete problems, such as the following DP-complete problem EXACT VERTEX COVER (XVC) that asks for a given undirected graph $G = (A,E)$ and a positive integer k whether $\tau(G) = k$, i.e., whether the size of a smallest vertex cover in G is exactly k.[1] Changing the question to whether $\tau(G) \leq k$ gives the well-known NP-complete VERTEX COVER problem (see, e.g., [7]).

[1] A *vertex cover* of an undirected graph $G = (A,E)$ is a subset $A' \subseteq A$ that contains at least one vertex of each edge. The size of a smallest vertex cover is denoted by $\tau(G)$. (As V is already used to denote voter lists, we will use A to denote vertex sets in graphs, although the common notation is V.)

3. The Margin of Victory and Destructive Bribery

In this section, we will give the formal definitions of the investigated problems and continue the work of Xia [24] by drawing further connections between the margin of victory and the complexity of destructive bribery. The margin of victory is defined as follows.

Definition 1 *For a given voting system $\mathscr{E}$ and a given $\mathscr{E}$ election (C,V), we define the* margin of victory *to be the smallest integer ℓ such that the winner set can be changed by changing ℓ votes in V, while the other votes remain unchanged. We will use the notation* $\text{MoV}(\mathscr{E},(C,V)) = \ell$.

Just as Xia [24], we will focus on the decision version of this problem denoted by $\mathscr{E}$-MARGIN OF VICTORY ($\mathscr{E}$-MoV) for a given voting system $\mathscr{E}$, that asks for an $\mathscr{E}$ election (C,V) and a positive integer k whether $\text{MoV}(\mathscr{E},(C,V)) \leq k$ holds.

The MoV problem is closely related to the standard bribery scenario in elections that was defined by Faliszewski et al. [5]. In particular, in $\mathscr{E}$-DESTRUCTIVE UNWEIGHTED BRIBERY ($\mathscr{E}$-DUB), we are given an $\mathscr{E}$ election (C,V), a designated candidate $p \in C$, and a positive integer k, and we ask whether it is possible to prevent p from being a unique $\mathscr{E}$ winner by bribing at most k voters (i.e., by changing their votes).

The above problem is defined in the so-called *unique-winner model*. By changing the question to whether the designated candidate can be prevented from being *an* $\mathscr{E}$ winner by bribing at most k voters, the problem would be defined in the so-called *nonunique-winner model* (a.k.a. the *co-winner model*). When analyzing the relationship between the bribery problem and the MoV problem, one has to pay close attention on whether the voting system at hand always selects unique winners or whether the winner set may contain more than one candidate. The following result, due to Xia [24], deals with the former type of voting rules and displays the close connection between the margin of victory and the standard bribery scenario in elections.

Proposition 2 (Xia [24]) *Let $\mathscr{E}$ be a voting system that always selects a unique winner of an election in deterministic polynomial time, and satisfies $\mathscr{E}$-*MoV $\neq \emptyset$.[2] *Then $\mathscr{E}$-*MoV *and $\mathscr{E}$-*DUB *are $\leq_{\text{m}}^{\text{p}}$-equivalent, i.e., $\mathscr{E}$-*MoV $\leq_{\text{m}}^{\text{p}}$ *$\mathscr{E}$-*DUB *and $\mathscr{E}$-*DUB $\leq_{\text{m}}^{\text{p}}$ *$\mathscr{E}$-*MoV.

For voting rules that may select more than one winner, however, we now show that the above equivalence does not hold in general, unless P could be shown to equal NP. Note that the voting rule we will construct for showing the following theorem is not neutral.[3] Whether there exists a neutral voting rule satisfying the same properties as the one we construct is an interesting open question.

Theorem 3 *There exists a voting system $\mathscr{K}$ such that $\mathscr{K}$-*DUB $\in P$ *but $\mathscr{K}$-*MoV *is* NP*-complete.*

Proof Sketch. Due to space constraints we only sketch the proof by providing the voting system $\mathscr{K}$ that always outputs at least two winners if there are at least two candi-

[2] As is common, we view a decision problem such as $\mathscr{E}$-MoV as the language of yes-instances.

[3] A voting rule is called *neutral* if the outcome does not depend on the candidates' naming, i.e., if any two candidates are swapped in each vote, the outcome changes accordingly.

dates. For an election (C,V) with $C = \{p\} \cup C'$, the winnerset in $\mathscr{K}$ is determined as follows:

$$\mathscr{K}(C,V) = \begin{cases} p & \text{if } C = \{p\} \\ \{p\} \cup cup(C',V) & \text{otherwise.} \end{cases}$$

It is easy to see that $\mathscr{K}$-DUB $\in$ P. NP-hardness of $\mathscr{K}$-MoV immediately follows from the NP-hardness of cup-MoV, which we will show in Theorem 5. ❑

4. Margin of Victory in Schulze and Cup Voting

Xia [24] established complexity results for the margin of victory problem for various voting rules, including all scoring protocols, STV, and Copeland elections. We now turn to the complexity of this problem in Schulze and cup elections.

Theorem 4 *For Schulze elections,* MoV *is* NP*-complete.*

Proof. NP-hardness directly follows from the NP-hardness result for Schulze-DUB shown by Parkes and Xia [12] and the fact that $\mathscr{E}$-DUB $\leq_{\mathrm{m}}^{\mathrm{p}}$ $\mathscr{E}$-MoV for each voting system $\mathscr{E}$ with $\mathscr{E}$-MoV $\neq \emptyset$. Membership of Schulze-MoV in NP is easy to see. ❑

Theorem 5 *For cup elections,* MoV *is* NP*-complete.*

Proof. This result follows from the NP-hardness of cup-DUB, which we will show by a reduction from the well-known NP-complete problem Vertex Cover (recall its definition from Section 2) using the so-called *UV technique* introduced by Faliszewski et al. [6]. To do so, let $G = (A,E)$ be an undirected graph with vertex set $A = \{a_1, a_2, \ldots, a_n\}$ and edge set $E = \{e_1, e_2, \ldots, e_m\}$, and let $k \in \mathbb{N}$. We construct the cup election (C,V) with $C = \{c,d\} \cup E \cup P \cup T$, where $P = \{p_1, p_2, \ldots, p_m\}$ and T is a set of dummy candidates that will be used to ensure that the voting tree is balanced (we will come to that later). Let $N_a = \{e \in E \mid e \cap \{a\} \neq \emptyset\}$ be the set of edges incident to vertex $a \in A$.

V contains $2m(n+k-3)+6n+6k-3$ voters whose preferences are listed in Table 1. When a set of candidates, say $Z \subseteq C$, is given in a voter's preference, then we assume that the candidates in Z are ordered with respect to a (tacitly assumed) fixed order, while $\overleftarrow{Z}$ denotes that the candidates are ordered in reverse. In particular, we fix the order of the candidates in P to be $p_1 > p_2 > \cdots > p_m$.

The dummy candidates in T are always positioned at the bottom of each voter's preference, so they lose every pairwise comparison to the candidates in $C \setminus T$. This implies that their position in the schedule is irrelevant, so we will omit them in Figure 1.

For the sake of readability and clarity, we will omit the dummy candidates in our further arguments, and we will use the voting tree and schedule shown in Figure 1. (To transform this tree into a complete binary tree (i.e., into a legal voting tree), the dummy candidates in T have to be added to the three subtrees with the roots d, c, and X, respectively.) Since the height of the tree is in $\mathscr{O}(\log m)$, we have in total a polynomial number of leaves, which ensures that the reduction is in fact polynomial-time computable.

Table 1. List of votes V for the proof of Theorem 5

#	Preference
one vote for each $a \in A$	$c\ d\ N_a\ P\ (E \setminus N_a)\ T$ $P\ c\ d\ (\overleftarrow{E \setminus N_a})\ \overleftarrow{N_a}\ T$
k votes	$c\ d\ P\ E\ T$ $c\ d\ P\ \overleftarrow{E}\ T$
$2(n+k-2)$ votes	$c\ E\ P\ d\ T$ $c\ \overleftarrow{E}\ P\ d\ T$
$n+k-3$ votes for each $i \in \{1,\dots,m\}$	$c\ (P \setminus \{p_i\})\ p_i\ e_i\ (E \setminus \{e_i\})\ d\ T$ $d\ (\overleftarrow{E \setminus \{e_i\}})\ p_i\ e_i\ (\overleftarrow{P \setminus \{p_i\}})\ c\ T$
one vote	$P\ c\ d\ E\ T$

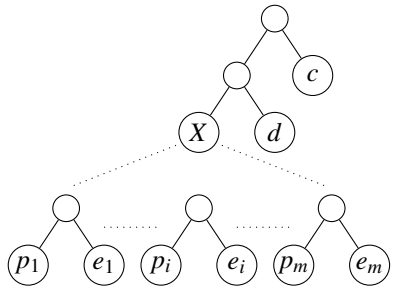


Figure 1. Voting tree of the cup election $(C \setminus T, V)$ without dummies

In this election, we have the following pairwise comparisons between the candidates in $C \setminus T$:

$$D_V(c,d) > 4k,\ D_V(c,P) = D_V(d,P) = 2k-1,\ D_V(c,E) = D_V(d,E) = 2n+2k+1,$$

$$D_V(p_i,p_j)\begin{cases} > 4k \text{ if } i<j \\ \leq 0 \text{ if } i \geq j, \end{cases} \qquad D_V(p_i,e_j) = \begin{cases} -2n-2k+5 & \text{if } i \neq j \\ -1 & \text{if } i = j. \end{cases}$$

Thus, c is the unique cup winner of this election. We claim that G has a vertex cover of size at most k if and only if c can be prevented from being a unique cup winner by changing at most k votes.

From left to right: Assume that $A' \subseteq A$ is a vertex cover of size k. Change the preferences of those k voters corresponding to A' in the first voter group from $c\,d\,N_a\,P\,(E \setminus N_a)\,T$ to $P\,c\,d\,N_a\,(E \setminus N_a)\,T$. Since A' is a vertex cover we have that due to these changes each $e_i \in E$ has one vote where she is positioned behind all candidates in P. So we have that each p_i wins her first pairwise comparison against e_i by one point. In the subelection corresponding to the subtree with root X (recall Figure 1), the relevant pairwise comparisons are among the candidates in P and due to the fixed ordering of these candidates in the votes, p_1 is the winner of this subelection. Both c and d have lost k votes in comparison to p_1 due to the bribe, so p_1 wins both pairwise comparisons and is thus the unique cup winner of this election. So c has been successfully prevented from winning.

From right to left: Assume that c can be prevented from being a unique winner by bribing at most k voters. Due to the scores only candidates from P have a chance to prevent c from being a unique winner, so the following has to hold for the bribed election: A candidate from P, say p_1, has to be the winner of the subelection corresponding to the subtree with root X and p_1 has to win the pairwise comparisons against both d and c. For the latter to hold, all k bribed votes have to have p_1 positioned behind d and c (before the bribe). For the former to hold, no candidate in E may win her first contest, which implies that every $p_i \in P$ has to win the pairwise comparison against the corresponding candidate $e_i \in E$. So the votes that are bribed also have to rank the candidates in E better than those in P before the bribe is conducted. With this we see that the k bribed votes have to be from the first voter group and that the vertices corresponding to these votes have to form a vertex cover of size k to ensure that each $e_i \in E$ loses the first pairwise comparison. ❑

5. Exact Margin of Victory in Schulze, Copeland, and Cup Voting

In this section, we present our complexity results for the exact variants of the margin of victory problem, which for a given voting system $\mathscr{E}$ is denoted by $\mathscr{E}$-EXACT MARGIN OF VICTORY ($\mathscr{E}$-XMOV) and asks for a given $\mathscr{E}$ election (C,V) and a positive integer k whether $\mathrm{MOV}(\mathscr{E},(C,V)) = k$. In particular, we will consider this problem for the two systems studied in the previous section, Schulze and cup, and also for Copeland$^\alpha$ voting. (Note that Xia [24] proved that Copeland$^\alpha$-MOV is NP-complete.) Due to space constraints we only provide a proof sketch for the following result in Schulze elections.

Theorem 6 *For Schulze elections,* XMOV *is* DP*-complete.*

Proof Sketch. For showing DP-hardness we provide a reduction from the DP-complete problem XVC. Let $G = (A,E)$ be an undirected graph with vertex set $A = \{a_1, a_2, \ldots, a_n\}$ and edge set $E = \{e_1, e_2, \ldots, e_m\}$, and let k be a positive integer. Without loss of generality, we assume that $6 \leq k \leq n$ and that $k-1 \bmod 5 = 0$. Let $U = E_1 \cup E_2 \cup E_3$ be the marked union of three copies of E, which are denoted by $E_i = \{e_{i1}, e_{i2}, \ldots, e_{im}\}$ for $i \in \{1,2,3\}$, and let $N_a = \{e_{ij} \mid e_j \cap \{a\} \neq \emptyset \text{ and } i \in \{1,2,3\}\}$ denote the set of all edges in U that are incident to vertex $a \in A$. We define the Schulze election (C,V), where $C = \{c,d,e,f,g,h,p\} \cup U$, and V is a list of $40n + 324k - 132$ voters, whose preferences are specified in Table 2. When a set of candidates $Z \subseteq C$ is given in a preference, we assume that the candidates in Z are ordered with respect to a (tacitly assumed) fixed order.

Figure 2 shows a subgraph of the weighted majority graph of this election that only contains edges relevant for the argumentation. Table 3 shows the weights of the relevant strongest paths in (C,V); hence, c is the unique Schulze winner in this election. We will elaborate on some useful properties of the constructed election: Since candidate c is the unique Schulze winner, the winner set can only be changed by achieving $P(c,x) \leq P(x,c)$ for at least one candidate $x \in C \setminus \{c\}$. Since $P(c,x) - 2k \geq 12k - 2k > \frac{12(k-1)}{5} + 2k \geq P(x,c) + 2k$ holds for all candidates $x \in C \setminus \{c,p\}$, only p can tie with c when no more than k votes can be changed. So it suffices to focus on the paths leading from c to p, and vice versa. From p to c, the only reasonable path is $((p,d),(d,e),(e,f),(f,g),(g,c))$.

Table 2. List of votes V in the proof of Theorem 6

#	Preference	Type
one vote for each $a \in A$	$h > c > g > f > e > N_a > p > d > (U \setminus N_a)$	1
	$c > g > e > f > d > N_a > p > (U \setminus N_a) > h$	1
	$h > c > g > f > d > e > N_a > p > (U \setminus N_a)$	1
	$c > f > g > e > d > N_a > p > (U \setminus N_a) > h$	1
	$h > g > c > f > e > d > N_a > p > (U \setminus N_a)$	1
n votes	$d > p > e > f > g > c > U > h$	2
	$h > p > d > f > e > g > c > U$	2
	$p > e > d > f > g > c > U > h$	2
	$h > p > d > e > g > f > c > U$	2
	$p > d > e > f > c > g > U > h$	2
$12(k-1)/10$ votes	$h > p > d > e > f > g > c > U$	2
	$p > d > e > f > g > c > U > h$	2
$3k-3+12(k-1)/10$ votes	$h > e > f > g > c > p > d > U$	3
	$d > c > g > f > e > p > U > h$	3
$6k+12(k-1)/10$ votes	$h > p > g > c > f > e > d > U$	2
	$c > d > e > f > g > p > U > h$	3
	$h > p > c > f > g > e > d > U$	2
	$g > d > e > f > c > p > U > h$	3
	$h > p > g > c > e > f > d > U$	2
	$f > d > e > c > g > p > U > h$	3
$6k$ votes	$c > h > g > e > f > p > d > U$	3
	$d > p > f > e > g > c > h > U$	2
	$h > g > c > e > f > p > d > U$	3
	$d > p > f > e > c > h > g > U$	2
$5n+41k-20$ votes	$h > d > c > g > E_1 > E_2 > p > E_3 > f > e$	4a
	$e > f > E_2 > E_1 > p > E_3 > g > c > d > h$	4b
	$h > d > c > f > E_1 > E_3 > p > E_2 > g > e$	5a
	$e > g > E_3 > E_1 > p > E_2 > f > c > d > h$	5b
	$h > d > c > e > E_2 > E_3 > p > E_1 > f > g$	6a
	$g > f > E_3 > E_2 > p > E_1 > e > c > d > h$	6b

Table 3. Weights of the strongest paths in (C,V) in the proof of Theorem 6

x	d	e	f	g	h	p	U
$P(c,x)$	$12k$	$12k$	$12k$	$12k$	$12k$	$6k-6$	$>12k$
$P(x,c)$	$\frac{12(k-1)}{5}$	$\frac{12(k-1)}{5}$	$\frac{12(k-1)}{5}$	$\frac{12(k-1)}{5}$	$\frac{12(k-1)}{5}$	$\frac{12(k-1)}{5}$	$\frac{12(k-1)}{5}$

We can argue that when at most k changes are allowed, we have that $P(p,c) \leq \frac{12(k-1)}{5} + \frac{8(k-1)}{5} = 4(k-1)$ holds in this new election. The following property can be shown for completing the proof:

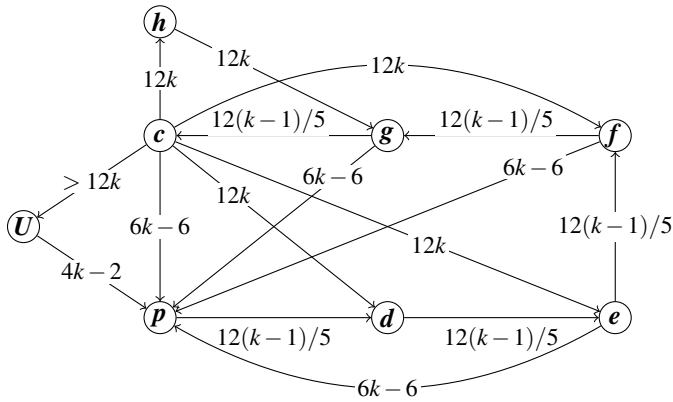


Figure 2. Subgraph of the WMG(C,V) of the Schulze election (C,V) in the proof of Theorem 6

$$\text{MoV}(\text{Schulze},(C,V))\begin{cases}=k-1 & \text{if } \tau(G)<k\\ =k & \text{if } \tau(G)=k\\ >k & \text{otherwise,}\end{cases} \tag{1}$$

where, recall, $\tau(G)$ denotes the size of a smallest vertex cover in G. We show the first case in (1) in detail: Assume that $\tau(G)<k$ and let $A'\subseteq A$ be a vertex cover in G of size $k-1$. For each vertex $a\in A$, there are five voters in V of type 1 (recall Table 2), which only differ in the ordering of the candidates $\{c,d,e,f,g,p\}$. If for each $a\in A'$ one of the five type-1 votes is changed such that $p>d>e>f>g>c>\cdots$ holds in this vote and these changes are carefully conducted while ensuring that all five votes are changed equally often, it can be achieved that $P(c,p)=4k-4=P(p,c)$. So we have that $\text{MoV}(\text{Schulze},(C,V))\leq k-1$. By changing at most $k-2$ votes in this election, one could achieve that $P(c,p)\geq 4k-2>4k-4\geq P(p,c)$, so c would remain the unique winner of the changed election. Thus we have that $\text{MoV}(\text{Schulze},(C,V))\geq k-1$, which gives $\text{MoV}(\text{Schulze},(C,V))=k-1$. ❑

For the other two voting systems, we have the same complexity results, and we omit their (similar) proofs due to space constraints.

Theorem 7 *For both cup and Copeland$^\alpha$,* XMoV *is* DP*-complete.*

6. Conclusions and Open Questions

Continuing the work of Xia [24], we have shown that the margin of victory problem is NP-complete for Schulze and cup elections, and its exact variant is DP-complete for Schulze, cup, and Copeland elections. For future research, it would be interesting to study the approximability of the margin of victory also for Schulze and cup. Furthermore the following variant of the MoV problem might be worthwile to analyze: Instead of counting the number of votes that have to be changed entirely to change an election's outcome, the overall number of swaps (or other changes in the votes depending on the voting system used) leading to a different winner set could be counted. This version would, e.g., model accidental errors in votes more naturally and allow a more fine-grained analysis.

Acknowledgments: This work was supported in part by DFG grant RO 1202/15-1.

References

[1] F. Brandt, V. Conitzer, and U. Endriss. Computational social choice. In G. Weiß, editor, *Multiagent Systems*, pages 213–283. MIT Press, 2013.

[2] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy I: Structural properties. *SIAM Journal on Computing*, 17(6):1232–1252, 1988.

[3] D. Cary. Estimating the margin of victory for instant-runoff voting. In *Website Proc. EVT/WOTE'11*, 2011.

[4] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? *Journal of the ACM*, 54(3):Article 14, 2007.

[5] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. How hard is bribery in elections. *Journal of Artificial Intelligence Research*, 35:485–532, 2009.

[6] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Llull and Copeland voting computationally resist bribery and constructive control. *Journal of Artificial Intelligence Research*, 35:275–341, 2009.

[7] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[8] T. Magrino, R. Rivest, E. Shen, and D. Wagner. Computing the margin of victory in IRV elections. In *Website Proc. EVT/WOTE'11*, 2011.

[9] N. Nguyen, T. Nguyen, M. Roos, and J. Rothe. Computational complexity and approximability of social welfare optimization in multiagent resource allocation. *Journal of Autonomous Agents and Multi-Agent Systems*, 28(2):256–289, 2014.

[10] L. Norden, A. Burstein, J. Hall, and M. Chen. *Post-Election Audits: Restoring Trust in Elections*. Brennan Center for Justice at the NYU School of Law and the Samuelson Law, Technology & Public Policy Clinic at the UC Berkeley School of Law (Boalt Hall), 2007.

[11] C. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). *Journal of Computer and System Sciences*, 28(2):244–259, 1984.

[12] D. Parkes and L. Xia. A complexity-of-strategic-behavior comparison between Schulze's rule and ranked pairs. In *Proc. AAAI'12*, pages 1429–1435. AAAI Press, 2012.

[13] A. Procaccia, J. Rosenschein, and G. Kaminka. On the robustness of preference aggregation in noisy environments. In *Proc. AAMAS'07*, pages 416–422. IFAAMAS, 2007.

[14] T. Riege and J. Rothe. Completeness in the boolean hierarchy: Exact-Four-Colorability, minimal graph uncolorability, and exact domatic number problems – a survey. *Journal of Universal Computer Science*, 12(5):551–578, 2006.

[15] A. Sarwate, S. Checkoway, and H. Shacham. Risk-limiting audits and the margin of victory for nonplurality elections. *Statistics, Politics and Policy*, pages 29–64, 2012.

[16] M. Schulze. A new monotonic, clone-independent, reversal symmetric, and Condorcet-consistent single-winner election method. *Social Choice and Welfare*, 36(2):267–303, 2011.

[17] D. Shiryaev, L. Yu, and E. Elkind. On elections with robust winners. In *Proc. AAMAS'13*, pages 415–422. IFAAMAS, 2013.

[18] P. Stark. Conservative statistical post-election audits. *Annals of Applied Statistics*, 2(2):435–776, 2008.

[19] P. Stark. A sharper discrepancy measure for post-election audits. *Annals of Applied Statistics*, 2(3):982–985, 2008.

[20] P. Stark. Risk-limiting postelection audits: Conservative-values from common probability inequalities. *IEEE Transactions on Information Forensics and Security*, 4(4):1005–1014, 2009.

[21] P. Stark. Super-simple simultaneous single-ballot risk-limiting audits. In *Website Proc. EVT/WOTE'10*, 2010.

[22] K. Wagner. More complicated questions about maxima and minima, and some closures of NP. *Theoretical Computer Science*, 51(1–2):53–80, 1987.

[23] S. Wolchok, E. Wustrow, J. Halderman, H. Prasad, A. Kankipati, S. Sakhamuri, V. Yagati, and R. Gonggrijp. Security analysis of India's electronic voting machines. In *Proc. ACM-CCS'10*, pages 1–14. ACM Press, 2010.

[24] L. Xia. Computing the margin of victory for various voting rules. In *Proc. ACM-EC'12*, pages 982–999. ACM Press, 2012.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-260

Electronic Tourist Guides: User-friendly Editing of Automatically Planned Routes

Richard SCHALLER [a]
[a] *E-mail: richard.schaller@fau.de*
AI Group, University of Erlangen-Nuremberg, Germany

Abstract. Sightseeing tourists on a city trip can be supported by an electronic tourist guide. More advanced systems generate personalized routes taking into account user preferences and traveling distances between sights. Recommenders can be used in order to estimate user's interest in the available sights. Typically recommendations are then fed into a planning algorithm that tries to determine a subset of sights and a suitable order. The aim being to include the most interesting sites for the user and possibly as many interesting sights as possible within a given set of constraints. Additional constraints can be considered such as a sight being a must-see for the user or opening hours of sights. At some point the generated route is shown to the user. If the user is not satisfied with the route he may want to change it. In most systems modifying the constraints or user preferences and then regenerating a completely new route is the only option to accomplish this task. More fine-grained control over route modifications is in typically unavailable.
In this paper we examine how sightseeing tourists can be supported in editing an automatically generated route. We discuss what is necessary for editing operations to meet user's expectations. We then show how six different route editing operations can be implemented via the idea of collapsing and expanding a route. Finally we present a potential route editing user interface for an electronic tourist guide. The presented interface makes disadvantageous route modifications obvious by providing instant feedback on how an operation affects the route.

1. Introduction

When planning a city trip tourists are faced with various tasks: a city and time frame has to be chosen, various traveling options and hotels have to be considered and finally the stay itself has to be planned. This last aspect is particularly challenging for trips to bigger cities with hundreds of sights, restaurants, shopping locations and various other tourist-related venues. These Points of Interest (POIs) tend to be geographically (e.g. throughout a city) and temporally (e.g. different opening hours during the day) dispersed. Tourists usually consider many different properties and contextual factors when deciding for which POIs to visit and in which order, i.e. the type, price, location and opening hours of the POI, the distance/transfer links to the POI and the personal interest in the POI but also general context factors such as time of day or weather [1]. An electronic tourist guide can assist visitors in finding optimal or near-optimal solutions to this multi-criteria optimization problem. In [2] three components of an electronic tourist guide are described: a recommendation phase that facilitates a user profile and POI data to generate a list of

personalized POI recommendations. In a second step, a planning algorithm determines an optimal subset and order of these POIs taking into account various constraints such as opening hours, traveling times or user restrictions such as start and end time of the route. As there might be user preferences or contextual factors not being modeled in the system it is necessary to have a third step which allows modifications of the generated route, like removal or insertion of a POI visit. Also during plan execution it is more than likely that visitors will need to re-plan due to external or internal causes, e.g. a POI being overcrowded or the tourist being exhausted. In [3] an electronic tourist guide was evaluated in a field trial: only 55% of the actually visited sights were contained in the route planed beforehand showing a need for route modifications on the move.
Most research performed on electronic tourist guides focuses on either the recommendation of POIs or on the planning of routes. The resulting routes are packed – as this is one of the objectives of the planning algorithm – making route modifications in most cases impossible without rendering the complete route infeasible. Consider, for example, adding a POI: if a gap in the route were big enough to insert a POI, the planning algorithm would already have done that. Most likely a visitor would expect to reduce the visit duration of other POIs to make room for the additional POI. On the other hand visitors would like to close gaps after removing POIs by expanding the visit of the remaining POIs. Additionally, moving POIs within a route might need to shorten visits as traveling times likely increase due to deriving from the optimal ordering.
In this paper we will focus on operations that can be performed on an already given route making it easy for visitors to modify routes to match their preferences:
We introduce the concept of collapsing the visit durations in a route to make room for further modifications. We furthermore, introduce the corresponding concept of expanding a route by increasing visit durations to fill gaps. We then show how user modifications can be implemented by making use of collapsing and expanding a route. Next we discuss removing, inserting and moving a POI, changing the visit duration of a POI and filling up a route with additional POIs. Finally we present a system where we implemented these ideas.

2. Related Work

Recommenders. Although this paper focuses on supporting route modification it is worthwhile to explore how recommendations can be obtained as they can be used for filling up a route with additional POIs. There are two main types of recommenders used in electronic tourist guides. These are content-based recommenders and collaborative filtering approaches. For content-based recommenders additional data for each POI and a user profile for each user have to be acquired. For example in [4] an ontology is used for categorizing each POI, e.g. as memorial. The user is then asked upon the first use of the system to specify his interest in these categories resulting in a detailed user profile. A semantic matching is then used to find those POIs that are most similar to the user's profile. Contrastingly, collaborative filtering approaches use other users' POI selections to learn which POIs are similar [5]: users with similar tastes are also likely to select the same POIs. Hybrid recommenders combine both approaches to improve the quality of recommendations [5,6]. An overview and comparison of the use of recommender systems in the field of electronic tourist guides is given in [1].

Planning. For planning purposes the optimization problem has to be formalized. [7] suggests to use the OP (Orienteering Problem) as the formal framework, a derivative of the Traveling Salesman Problem (TSP): a set of locations with an assigned score and information about traveling times between all locations is given. Additionally a starting point and time as well as a destination point and time is given. A solution of the OP is a route that contains a subset of the available locations and connects the starting point with the ending point. The optimal solution is the route that maximizes the sum of the collected scores. The OP can be extend to consider various additional constraints such as opening hours (Orienteering Problem with Time Windows) or to plan for multiple days (Team Orienteering Problem). For solving the OP both exact [8] and heuristic [4] approaches exist. In [7] an Iterative Local Search is used to solve the OP: a route is created by iteratively adding and removing POIs. Choosing which POI to add considers the score of the POI and the additional required time caused by the POI insertion. Adding POIs is done until no further POIs can be added and the algorithm is stuck in a (local) optima. Then removing of POIs is performed to escape the local optima. Under all circumstances the route is feasible, meaning start and end time constraints, traveling times and visit durations are considered and additional constraints such as opening hours are respected.

Electronic Tourist Guides. Various research projects have developed prototypical or complete electronic tourist guides: The CT-Planner described in [9] uses a content-based recommender to calculate scores for all POIs. For route generation a heuristic approximation algorithm is used. To obtain a user profile a user can specify his interests directly or indirectly by choosing between different routes. The latter are generated by using the recommender with a slightly modified user profile. In case the user decides for one of these alternatives the user profile is updated accordingly. Furthermore the user can influence the route generation by selecting certain POIs as must-haves or don'ts. However, there is no possibility to edit the generated route in detail. The system developed in [2] uses a content-based recommender to choose POIs but also let users specify which POIs are of interest to them. The planning is based on an Iterative Local Search algorithm. To customize the generated route various operations such as adding, removing or moving a POI are provided. These operations shift beginning times of all POIs after the editing position accordingly. It is possible for the user to make modifications that violate some of the restrictions. From the description given in [2] it does not become clear whether this applies only to user specified restrictions such as the end time or to all kinds of restrictions. Violating POI restrictions such as opening hours would render routes infeasible. Thus it is plausible that such modifications are forbidden. Permitted modification will shift visits beyond the specified end time as needed, effectively removing this constraint. To our knowledge, visit durations are not adjusted.

3. User-friendly editing operations

Considering the systems presented in the previous section most do either provide no route modifications or do so via recalculating a route under modified constraints. A fine-grained control of routes is only provided by [2] where the route end time constraint is lifted. Depending on the type of POIs this lifting might not be enough to yield still feasible routes. Consider for example POIs with very limited opening hours where shifting results in missing the opening hours. The user is either left behind with no explanation

why his operation is not permitted or a complex explanation – probably involving multiple constraints – must be generated and communicated.
We instead suppose to lift a different constraint: the visit duration of POIs. In most systems its value is either fixed per POI or estimated in advance for each user. We argue that the visit duration is a soft constraint from a user's point of view and the user is flexible regarding the exact visit duration: routes do not suddenly become infeasible if a museum visit is scheduled for a few minutes less. Of course further editing might reduce the visit duration even more and make the route unrealistic. However, our approach has the advantage that the user can see how the route slowing becomes more and more infeasible and receives feedback regarding why his editing operations are unfavorable. Thus we suggest to adjust the visit durations of other POIs in order to allow editing operations under most circumstances. The only exception where editing operations are still forbidden is when visit durations cannot be shortened further or route independent conflicts make the operation fail, e.g. visiting two POIs at the same time at different locations.
As the visit duration becomes very flexible and time can be shuffled arbitrary between POIs we make some assumptions regarding how the system should behave when performing route modifications:

- The route should always be feasible meaning all constraints are met (except the visit duration constraint): traveling times are considered, opening hours are respected, etc.
- The order of the POI remains unchanged except for the POI that is involved in the modification.
- If there are gaps in the route they should be reduced as much as possible.
- With each POI a minimum visit duration is associated which should be respected unless this results in a route modification becoming inexecutable. In this case the visit duration might be reduced further still.
- The time surplus caused by gaps should be distributed "fairly" between available POIs, meaning that time is distributed equally among all POIs. If a POI's visit duration cannot be extended any further the remaining time is distributed among the other POIs.
- Also if time is needed to perform an operation, visit duration of all POIs should be reduced in a "fair" manner. As minimum visit durations should be respected, time should be taken first from those POIs which are scheduled for a longer stay. We consider the minimum visit duration as sufficient and any additional time spent (=extra visit duration[1]) as not contributing to visitor's experience. Nevertheless we want to keep the extra visit time of all POIs equal which leads to first shortening that POI which has the most extra visit duration.

With these assumptions we can now redefine the route operations from [2]: *Remove a POI*, *Insert a POI at a specific position*, *Insert a POI at its best position* and *Move a POI*. Additionally we allow the user to *Change the visit duration of a POI*. In case the user removes multiple POIs he might like to *Fill-Up the route with POIs* without specifying the POIs to add but instead let the system decide which POIs to use.

[1] visit duration = minimum visit duration + extra visit duration

4. Collapsing and Expanding a Route

The different editing operations that we are going to implement either consume time or leave additional time after they were performed. For the former we want to provide as much flexibility as possible which means that we want to reduce the time spent on POIs as much as possible. We call this process of transforming a given route into a route, where every POI is only visited for no longer than its respective minimum visit duration, *collapsing a route*. The delta between the old visit duration and the new visit duration is remembered. The following pseudo code shows this function:

```
Collapse(route):
  FORALL p in route:
    diff = max(0, p.duration - p.minDuration)
    p.delta += diff
    newDuration = p.duration - diff
    Change duration of p to newDuration
```

Of course changing the visit duration of a POI results in shifting all subsequent POI visits accordingly. Shifting POI visits is a crucial part of the Iterative Local Search approach for solving the OP. Making it efficient and also taking into account restrictions such as opening hours is described in [7].

As mentioned before we make the assumption that the minimum visit duration is respected but there might be situations where we allow shorter visit durations. We therefore define a *total collapsing of a route*, which reduces the visit duration of each POI to a small fixed value – 1 min in our case[2] – as follows:

```
TotallyCollapse(route):
  FORALL p in route:
    diff = max(0, p.duration - 1)
    p.delta += diff
    newDuration = p.duration - diff
    Change duration of p to newDuration
```

The inverse operation of collapsing a route is *expanding a route*. To restore a previously collapsed route we first use the saved delta and distribute the available time between those POI visits that were previously shortened (delta > 0). If there are still gaps in the route the remaining time is distributed equal among all POIs. Thus expanding is split up into two steps, which differ only in the POIs being affected:

```
Expand(route):
  ExpandHelper(route,useOnlyDelta=True)
  ExpandHelper(route,useOnlyDelta=False)
```

Even though it is possible to calculate for each POI by how much its visit duration has to be extended and update them at a stroke we instead opted for incrementally expanding a route. This simplifies the process and makes it easy to take account of additional constraints, e.g. opening hours. Expanding is performed by successively increasing the visit duration of each POI one after another by a small amount – we use a step size of 1 min. For every expansion of a POI visit its corresponding delta is reduced. In case a POI visit becomes stuck and cannot be expanded further – again opening hours being a potential cause – we skip this POI and continue with the next one. After all POIs have been updated we start all over again until no further expansions are possible.

The following pseudo code shows how expanding a route is performed. Note the use of `useOnlyDelta` to select whether all POIs should be affected or only those with a saved delta:

[2]We choose 1 min as we do not want to confuse visitors with 0 min visits

```
ExpandHelper(route, useOnlyDelta):
  DO:
    expanded = False
    FORALL p in route:
      IF useOnlyDelta AND p.delta == 0:
        CONTINUE
      IF duration increase by 1 feasible for p:
        newDuration = p.duration + 1
        Change duration of p to newDuration
        p.delta = max(0, p.delta-1)
        expanded = True
  WHILE expanded
```

One caveat of the presented method is that POIs towards the start of a route are considered first. In case only a small amount of time becomes available it is given to those POIs. If this happens repeatedly instead of all time becoming available at once[3], time is not fairly split between those POIs towards the beginning and those towards the end of the route. To remedy this problem one may remember at which POI the last expansion stopped and continue from there instead of the first POI.

Consider what happens if a route is collapsed, then some operations are performed on it and then it is expanded again. There are three cases of how available time changes depending on the performed operations: a) Same amount of time is used: with the first run of `ExpandHelper` the original visit durations are restored. b) Some time was freed up: again the original visit durations are restored as above. But there is still some time left, which is distributed equally by the second run of `ExpandHelper`. c) Some time was used up: with the first run of `ExpandHelper` the original visit durations could not be restored completely, but the available time is again distributed equally. This means that POI visits that were reduced by a larger amount during collapsing than others are those whose time is partially taken away to account for the used up time. They are not fully expanded any further. The last case also means that delta remains > 0 for some of the POI visits. In order to not affect further editing operations it is necessary to reset it.

Of course expanding a route can also be used to reduce gaps in a newly generated route: planning algorithms may schedule POIs with their respective minimum visit duration and the resulting route is then expanded afterwards.

5. Route Editing Operations

All editing operations described in this section follow a general scheme: (totally) collapsing the route, performing the actual edit and finally expanding the route. Sometimes slight variations of this scheme are used to realize the provided editing operations.

Remove POI. As removing POIs can only result in time being freed up, collapsing the route can be skipped and expanding the route will close the resulting gap.

Insert POI at a specific position. For insertion of an POI at a specific position we first totally collapse the route, then insert the POI and finally expand the route:

```
Insert(route, poi, position):
  ResetDelta(route)
  TotallyCollapse(route)
  Insert poi at position
  Expand(route)
```

Totally collapsing the route might result in POI visits dropping below their minimum

[3] i.e. when a user can modify the visit duration interactively via a slider interface (see Section 6)

visit duration but this is either fixed when expanding the route (because other POIs' extra visit duration can be used) or it is necessary and there is no way around it as time has to be freed up for the added POI.

Insert POI at its best position. If the system should determine the best insertion position for a POI, we initially want to consider only positions which would not result in other POI visits dropping below their minimum visit duration. Thus we start with collapsing the route to their POIs minimum visit duration. Then we try to insert the POI. If this is not possible we must shorten visits below their minimum visit duration with a total collapse of the route and try the insertion of the POI again. In any case the route is expanded afterwards:

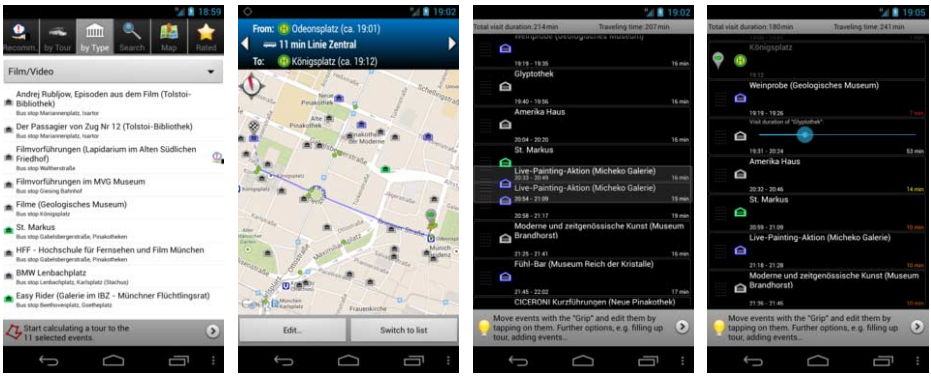
```
Insert(route, poi)
  ResetDelta(route)
  Collapse(route)
  p = find best insertion position for poi
  IF p is valid:
    Insert poi at p
  ELSE:
    TotallyCollapse(route)
    p = find best insertion position for poi
    IF p is valid:
      Insert poi at p
  Expand(route)
```

Move POI. When moving a POI we adhere to the general scheme of total collapsing, performing the move and expanding.

POI visit duration change. Changing the duration of a visit depends on whether it is shrunk or extended. In case of a shrunk collapsing the route is not necessary because no additional time is needed. As the duration of the affected POI is set directly it must be excluded from the following expanding of the route (lock). Otherwise its visit duration might get re-increased. In case a visit duration is extended we first totally collapse the route. Then we change the visit duration. If the requested new duration of the POI is not feasible we only change it to the maximum feasible value. Again we have to make sure that the expanding of the route will leave out the modified POI:

```
ChangeDuration(route, position, newDuration):
  poi = route[position]
  oldDuration = poi.duration
  IF newDuration < oldDuration:
    ResetDelta(route)
    Change duration of poi to newDuration
    lock poi.duration
    Expand(route)
    unlock poi.duration
  ELSE IF newDuration > oldDuration:
    ResetDelta(route)
    TotallyCollapse(route)
    maxDuration = max feasible duration of poi
    d = min(maxDuration, newDuration)
    Change duration of poi to d
    lock poi.duration
    Expand(route)
    unlock poi.duration
```

Fill-up route. Routes with larger gaps might be filled up by letting the system decide which POIs to add. The systems choice might be restricted to a set of POIs – a candidate set. Recommenders could be used to determine the set and/or to weight POIs. For filling up a route we first collapse all visits to their minimum visit duration. Then we perform one step of the Iterative Local Search algorithm for the OP described in [7]: we iteratively



(a) (b) (c) (d)

Figure 1. (a) Browsing through the available events by different means. (b) Following a generated route on the map. (c) Editing the route by moving an event. (d) Editing the route by changing the visit duration.

insert one POI after another until the route is too packed to add further POIs. In each iteration we determine the best candidate and its best position. We then add this POI at that position to the route and remove it from the candidate set. Afterwards we expand the route again to close any remaining gaps:

```
FillUp(route, poiSet)
  ResetDelta(route)
  Collapse(route)
  DO:
    insert best POI of poiSet at best position
    IF route was changed
      remove inserted POI from poiSet
  WHILE route was changed
  Expand(route)
```

Our local search approach on filling up a route results in a local optimum to be returned which might be unfortunate. However, in our experience this seems to be less problematic as long as the route to be filled-up is not empty or near empty: the existing POIs reduce the search space, set a general frame for the resulting route and thus make sure that the generated plan is plausible.

6. Route Editing User Interface

The editing operations described in the previous section can be implemented via a user interface designed to assist visitors in modifying a generated route. We implemented an electronic tourist guide for a special kind of tourism: visiting a distributed event. A distributed event [10] is a collection of smaller, single events occurring at approximately the same time and conforming to one overarching theme. Well known examples include the Cannes International Film Festival, the Edinburgh Festival Fringe, and Montreal International Jazz Festival. We focus in this paper on the Long Night of Munich Museums. What many of these events have in common is that they have huge number of diverse sub-events that are geographically and temporally dispersed. Thus visitors of distributed events are facing very similar problems to those visiting a city for sightseeing purposes. One main difference are the special shuttle buses that are provided and that connect the different events. In the developed Android app we provide assistance for three components of electronic tourist guides mentioned in [2]:

In a first step we determine a set of POIs of potential interest to the visitor. For this purpose we use a recommender system [5] but do not feed the results into the route planning algorithm. Instead the recommendations are shown to the user to allow him to decide upon the events he is interested in. This also makes it possible to provide users other means of accessing events [11]. Each tab in Figure 1a provides one of these options: the user can browse events organized geographically by their position on bus routes, by type, by searching for their name/description or by selecting them on a map. The "Rated" tab shows a list of already selected events.
In a next step we generate routes containing as many events as possible from the set of selected events. This is done with an Iterative Local Search algorithm similar to [7]. The resulting route presented to the user as a list or on a map (see Figure 1b).
Finally when the user decides to modify the route we support him on a dedicated editing screen (see Figure 1c and Figure 1d): all the editing operations described in Section 5 are available: when adding events the user is directed to the already known event browser (see Figure 1a) to choose an event. Moving an event is performed by drag and drop as depicted in Figure 1c. During the dragging of an event the route is continuously updated to reflect the current position of the event in the route. For changing the event visit duration we use a slider interface as shown in Figure 1d: dragging the slider to the right instantaneously increases the visit duration of the selected event and decreases those of the other events. As for each slider movement the complete route has to be collapsed and expanded we experienced some performance problems on older low-end smartphones. To solve these problems and provide a responsive user interface we increased the step size to 5 min when expanding the route. This only partially expands the route and would leave gaps of up to 4 min which we close by expanding again with a 1 min step size.
At the top of the editing screen the total time spent at events and the total traveling time is shown. This helps users in judging the effects of an editing operation on the route as a whole. Additionally event visits dropping below the minimum visit duration are visually highlighted via color depending on how much they fall short of the desired visit duration. This makes it easy for the user to realize that this constraint is violated and might encourage him to make further route modifications to remedy this problem.

7. Conclusion and Future Work

In this paper we have provided insights into how sightseeing tourists could be supported in editing an automatically generated route. We first looked into why it is desirable to give users the opportunity to modify a route instead of focusing on improvements to the POI recommender or the route planning algorithms. We also gave an overview on how current systems approach this need showing that all but one system rely on re-planning. Only that system provides fine-grained control on route modifications to the user but ignores the user-specified route end time. We argued that this is neither sufficient to allow certain route modifications in case POIs have opening hours nor do users expect the system to dismiss their input. Thus we next made up guidelines into how an editing system should behave to meet users' expectations such as a fair distribution of time surplus. We finally set on six operations we want the system to support. To define these we established and defined the concept of collapsing and expanding a route. Collapsing is reducing all POI visits to the bare minimum while expanding a route is the inverse

operation and furthermore closes all remaining gaps in the route. With the help of these we then provided solutions for the six operations which adhere to the guidelines made up earlier. We showed that most operations consist of the three steps collapsing the route, performing edit operation and then expanding the route again. Finally we looked into an assistance system for distributed events and especially into the route editing user interface that uses the operations developed in this paper. The interface provides instant feedback on how an operation affects the route making disadvantageous route modifications obvious while not hindering users in performing the modifications they envisioned.
Even though some of the operations make use of ideas used by planning algorithms, the presented solutions are independent of these and can be used in conjunction with any route generation algorithm. Nevertheless, a tighter integration between planning and route modifications is imaginable: the fill-up algorithm presented in this paper is a first step in this direction. More advanced route improvement strategies could recommend advantageous editing operation sequences such as: "removing POI A, switching POI B and C and then adding POI D, E and F would save you a lot of traveling and would increase the total time spend at POIs." This would show the user a way back from a customized route to an optimized route.
In the near future we plan to evaluate how the editing interface is used in a field trial. We are interested in how often the available operations are used, what events they are used on and in which way the route is modified. We plan to use metrics, such as the topical diversity of route events, to measure how routes before and after the edit differ from each other. This might gain insights into what properties a route generation algorithm should take into account. If user-made route improvements can be measured it might also be possible to use these metrics during the planning phase to judge different route options.

References

[1] Katerina Kabassi. Personalizing recommendations for tourists. *Telematics and Informatics*, 27(1):51–66, 2010.

[2] Ander Garcia, Olatz Arbelaitz, Maria Teresa Linaza, Pieter Vansteenwegen, and Wouter Souffriau. Personalized tourist route generation. In *Current Trends in Web Engineering*, pages 486–497. Springer, 2010.

[3] Ronny Kramer, Marko Modsching, Klaus Hagen, and Ulrike Gretzel. Behavioural impacts of mobile tour guides. *ENTER*, pages 109–118, 2007.

[4] Ronny Kramer, Marko Modsching, and Klaus Ten Hagen. A city guide agent creating and adapting individual sightseeing tours based on field trial results. *International Journal of Computational Intelligence Research*, 2(2):191–206, 2006.

[5] Richard Schaller, Morgan Harvey, and David Elsweiler. RecSys for distributed events: Investigating the influence of recommendations on visitor plans. In *Proc. of SIGIR*, 2013.

[6] Eui-young Kang, Hanil Kim, and Jungwon Cho. Personalization method for tourist point of interest (POI) recommendation. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 392–400. Springer, 2006.

[7] Pieter Vansteenwegen. Planning in tourism and public transportation. *4OR*, 7(3):293–296, 2009.

[8] Marcus Poggi, Henrique Viana, and Eduardo Uchoa. The team orienteering problem: Formulations and branch-cut and price. In *10th Workshop on ATMOS*, page 142, 2010.

[9] Yohei Kurata. CT-Planner2: More flexible and interactive assistance for day tour planning. In *Information and Communication Technologies in Tourism 2011*, pages 25–37. Springer, 2011.

[10] Richard Schaller. Planning and navigational assistance for distributed events. In *Proceedings of the 2nd Workshop on Context Aware Intelligent Assistance*, 2011.

[11] Richard Schaller, Morgan Harvey, and David Elsweiler. Entertainment on the go: finding things to do and see while visiting distributed events. In *Proc. IIiX*, pages 90–99, 2012.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-270

A Cost-Based Relaxed Planning Graph Heuristic for Enhanced Metric Sensitivity

Michal SROKA [a], Derek LONG [a]
[a] *name.surname@kcl.ac.uk*

Abstract. Most applications of planning depend on finding high quality solutions. The quality is evaluated in terms of user defined metric function. We discuss the current state-of-the-art for finding good quality solutions, and its limitations. We determine which planners are metric sensitive and are driven by cost. Following that we present a novel metric sensitivity heuristic using a modified version of the relaxed planning graph. The proposed heuristic helps in generating plans in response to the change of the metrics.

Keywords. planning, multi-objective, metrics, metric sensitive

Introduction

Practical applications of planning depend on finding good quality plans, where the quality is generally not measured simply by the number of actions in the plan. Instead, it is usual for the quality of a plan to be measured by costs associated with different actions, which reflect consumption of resources: typically time, energy or money. PDDL2.1 [4] introduced the extension that allows a user to specify the plan metric by which the quality of a plan is to be measured, but it is still the case that most modern planners ignore this specification and simply focus on generating short plans. Most benchmark domains exhibit a close correlation between plan length and plan cost, so the fact that planners ignore the cost is obscured in empirical evaluations of performance.

In this paper, we explore the property of metric sensitivity [11], which means that a planner responds directly to the plan metric. More specifically, we consider examples of problems in which the optimal solution can vary significantly as the metric changes, and then examine the small set of modern planners that are responsive to the plan metric. User-specified plan metric is used to evaluate the plan quality. The use of metric fluents may cause the action cost to vary depending on the state in which they are applied. A planner can mitigate the cost by appropriately preparing the state.

After a short example we briefly review the state-of-the-art of planning with metric functions, a description of domains which offer trade-offs between resources to achieve the goal. A selection of current state-of-the-art planners, together with evaluation of their metric sensitivity. We conclude that very few demonstrate metric sensitivity. In Section 3, we introduce a novel method for calculating a metric sensitive heuristic function using a cost-based Relaxed Planning Graph. An implementation of the method is then presented using a novel compilation from a non-temporal cost domain to a temporal domain. We conclude with a comparison between our method and relevant state-of-the-art planner.

1. Example

We now present a simple concrete example problem that illustrates some of the issues involved in finding high quality plans. Suppose we want to transport two packages from location $L0$ to location $L5$. We have two drivers and three vehicles available, one electric, *Te*1, and two diesel, *Tf*1 and *Tf*2. The amount of resource used by a vehicle is equal to the distance driven multiplied by the *square* of the loaded truck weight (number of packages plus one) using metric fluents this is $(resource\text{-}used) = (distance\ ?l1\ ?l2) * (load\ ?t)^2$ where resource can be diesel or electricity. This problem is represented in Figure 1. The following plan metric is used for evaluation: $(minimize(+(*A(electricity\text{-}used))(*B(fuel\text{-}used))))$. Where A and B vary, to check how planner responds to that change.

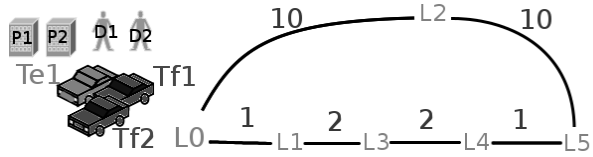


Figure 1. A simple problem with different vehicle types to transport packages $P1$ and $P2$ to $L5$.

Plans solving this problem are summarised below:

1) Load both packages into *Tf*1 and then drive them to $L5$ via $L2$ using driver $D1$. Cost: 180 diesel, 0 electric, length: 6.	2) Load both packages into *Tf*1 and drive to $L5$ via $L1$, $L3$ and $L4$, using $D1$. Cost: 54 diesel, 0 electric, length: 9.
3) Load both packages into *Te*1 and drive to $L5$ via $L1$, $L3$ and $L4$, using $D1$. Cost: 0 diesel, 54 electric, length: 9.	4) Load one package into *Tf*1 and one into *Tf*2 and drive both to $L5$ via $L1$. Cost: 48 diesel, 0 electric, length: 14.

Plans 1, 2, 3 and 4 are all different, both qualitatively and quantitatively. It is clear that the optimal cost plan, under any combination of metric fluents, will involve using the shorter path. The fact that this uses more actions is a problem for many planners, which attempt to minimise plan length as a proxy for the plan quality.

Using a single plan metric combining metric fluents like fuel cost and electricity cost with respective weights of 9 and 8, the optimal plan uses one diesel truck and the electric truck, each carrying one package (cost 24 diesel and 24 electricity, with total weighted cost of 408, while both plans 3 and 4 have weighted cost of 432). This example illustrates how interesting choices arise according to the trade-offs between resources being used.

2. Background

For generation of good quality plans under different plan metrics we require the planner to be metric sensitive [11]. A planner is metric sensitive if, presented with two different plan metrics (cost), m_1 and m_2, it produces different plans, π_1 and π_2 for the same problem instance, such that $m_1(\pi_1) < m_1(\pi_2)$ and $m_2(\pi_1) > m_2(\pi_2)$. In practice, a metric sensitive planner will not always be able to find two different plans that are each better under their respective plan metrics. In their specification of PDDL2.1, Fox and Long [4] introduced plan metrics, which are functions of the metric fluent parameters of a planning problem used to evaluate the cost or reward of a solution plan. For example:

```
(:metric minimize (+(energy)(+(*5 (labour))(*4 (pollution)))))
```

Radzi [9] distinguishes different interactions between plan lengths and plan metrics. In the following we use π and $\pi;\sigma$ to stand for executable sequences of actions, where the latter is a concatenation of two sub-sequences, $c(\pi)$ as the cost of π and $|\pi|$ as the length of π. We say a plan metric, c, is *strictly length-correlated* if $\forall\pi_1\pi_2 \cdot c(\pi_1) \leq c(\pi_2) \rightarrow |\pi_1| \leq |\pi_2|$, *monotonic* if $\forall\pi\sigma \cdot c(\pi) \leq c(\pi;\sigma)$, *non-monotonic* if $\exists\pi\sigma \cdot c(\pi) > c(\pi;\sigma)$. While most current planners optimise quality only when the plan metric is strictly length-correlated, only metric sensitive planners can effectively deal with monotonic metrics. There are currently no planners, including our own, that deal effectively with non-monotonic metrics.

Keyder and Geffner [7] present one approach to solving domains with monotonic plan metric. Their heuristic is calculated backward from the goal taking the cheapest achievers of each open facts, where the achievers are taken from the RPG constructed. This gives the cheapest relaxed plan which is found when constructing the RPG, however, it is still prone to bias demonstrated in the example as the actions which does not appear in the RPG, like driving via l1, are not considered in the construction of the relaxed plan.

2.1. Domains

Current benchmark planning domains do not usually provide us with interesting plan metric functions that are unrelated to the plan makespan. In our experiments we use domains which contain the possibility of trade-off between various resources, introduced in [9], Bread and Production. They contain the property that the length of the plan does not correspond with the quality and, therefore, this property forces the planner to look for a better rather than shorter solutions in order to increase the quality. We also introduce a modified Driverlog domain [8], decoupling the plan length and quality.

The Bread domain describes a process of baking bread and buns. We start with flour which we turn into a mix. The mix is used to create a dough. Dough is either created by hand or using a machine. Using the machine increases energy consumption and the machine needs cleaning, but can create twice more dough comparing to doing it by hand. The next stage is to make a bun or a bread from the dough. From the same amount of dough we can form either two loaves of bread or five buns. They can be baked using either an electric oven or on charcoal. An electric oven uses one unit of energy and charcoal increases the pollution by one unit. An electric oven can bake ten buns or four loaves while charcoal only two buns or two loaves. Our plan metric function is composed of weighted sum of the following metric fluents: energy, labour, pollution.

The aim in the Production domain is to obtain a certain amount of materials and ready products in stock. There are various ways of creating the same product or obtaining materials. The planner has flexibility to use various methods to arrive at the same goal which differ in labour, hazard or machine-cost values.

The Driverlog domain [8] used in experiments is the standard benchmark but differentiates electric and diesel trucks which creates a trade off between diesel and electricity consumption used for transportation of packages. Driverlog_metric was further modified by an addition of multiple shortcuts for driving between cities where, if used, the resource consumption is much smaller but the number of drive actions is greater. This change is a good test for planners which are truly metric sensitive. In our experiments we have used two versions of Driverlog domain: Basic, where no changes are made to

the infrastructure, metric fluents, and plan metrics, where three locations, $s0$, $s1$ and $s2$, are connected by very short routes with two intermediate steps.

2.2. Satisficing planners

Satisficing planners, due to their speed, offer a more promising approach to solving larger problem instances. We briefly present some state-of-the-art planners which are capable of reasoning with plan metrics as specified in PDDL2.1 [4]. Many satisficing planners can generate high quality plans for domains containing metrics. Some of them, like Lama [10]; aim at generating high quality solutions, however, they typically do not support :FLUENTS. The only metrics which are allowed are the metrics which affect the (TOTAL-COST) function which is assumed to be the cost of an action. Our work focuses on providing the flexibility for the user of the planning system to define functions against which the plan is evaluated. Therefore these approaches are not applicable, since the planner must generate good quality plans depending on the metric. The candidates which are promising for exhibiting metric sensitive behaviour are presented below.

MetricFF [6] is a very successful domain independent planner working with PDDL2.1. For each state which it evaluates it solves a relaxed version of the planning problem to obtain the cost of arriving to the goal state. The most recent version is capable of handling metric fluents and optimising a cost function, so it is a good candidate for a metric sensitive planner.

LPG [5], is a domain independent, local search, stochastic planner. It creates its search space based on a graph with interleaved proposition and action layers. Its heuristic consists of two elements, estimating the expected search cost and the expected execution cost to complete the plan from an evaluated search state, where search cost is the cost to resolve all the flaws in the current search state and execution cost is the estimated total cost of executing actions in the plan. Execution cost therefore represents plan quality. There are two weights on these two components, which allow trade-off between finding solutions quickly or searching for good quality solutions.

POPF2 [2] is a deterministic temporal planner that attempts to find minimum makespan plans. It uses a Simple Temporal Network (STN) to handle temporal constraints between actions and schedule them in a feasible way. It handles metric fluents, as for PDDL2.1, and therefore we believe it is a good candidate.

LPRPG [1] uses relaxed planning graph (RPG) heuristics combined with linear programming (LP) methods. It solves a number of LP for every decision it makes to calculate bounds on resources and to improve its numeric reasoning. Doing this gives LPRPG more precise information about bounds on resources than other planners and therefore is designed for use in domains with numeric resource.

2.3. Metric sensitivity test

Here we present results of the planners discussed in Section 2.2. Each planner is presented with the same problem, in eleven different versions, where each version is only different in the plan metric function to be minimised. An instance of a problem from the modified Driverlog domain, described in Section 2.1, is used. In order to show metric sensitivity, we expect the planners to generate different solutions. This can be achieved by using electric or diesel vehicles, depending on the cost of each of the resources (diesel

and electricity). The plan metric function to minimise in each of the examples is ((10-i) * (fuel-used) + i * (electricity-used)), for i=1,2,...,10. Figure 2 presents the results. It is clear that the only planner which generates different results is LPG.

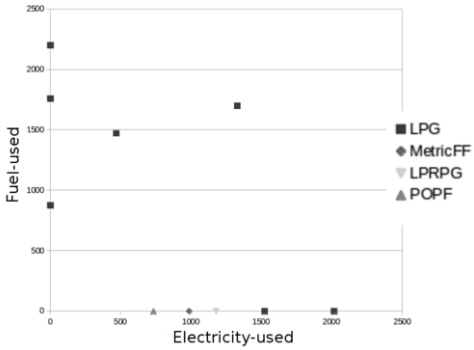


Figure 2. Results for running each of the planners 11 times on the same problem file, but with different plan metric function to minimise. For MetricFF, POPF2 and LPRPG marks on the figure represent 11 identical plans generated for 11 different plan metrics.

POPF2, LPRPG and MetricFF perform poorly in this test and do not exhibit metric sensitive behaviour. We attribute this fact to the RPG-based heuristic which prefers plans that achieve the goal in fewer action layers, instead of cheaper in terms of plan metrics.

LPG performed well by generating different plans for different plan metric function for the same problem. It is therefore our choice for later comparison.

POPF2 will produce the same plan for a given problem instance, regardless of the plan metric presented to it, since it does not respond to the plan metric. This does not seem a very promising planner to consider in the context of the current work, but we will show that its approach to optimising makespan can be harnessed to allow it to be metric sensitive in solving non-temporal problems.

3. A cost-based relaxed planning graph

Most current planners use Relaxed Planning Graph (RPG) as the base for their heuristics. This approach proved to provide a very good heuristic estimates efficiently for STRIPS domains. When dealing with numeric domains where the plan quality is evaluated using metric fluents RPG based approaches suffer from its bias towards shorter plans. Although for strictly length correlated plan metrics, this is not a problem as shorter plans are also plans of better quality, when faced with real life problems who are commonly monotonic or non-monotonic, this approach performs poorly. We propose a novel approach to constructing a relaxed planning graph based on cost. This novel approach aims at preferring cheaper, in terms of the cost, rather than shorter plans.

Figure 3 demonstrates how an example cost-RPG is constructed for the problem described in Figure 1 and metric function $(:\ metricminimize(+(*1(electricity - used))(+(*2(fuel - used))(*1(walked)))))$ to minimise. Construction of cost-RPG layers is done in the following steps:

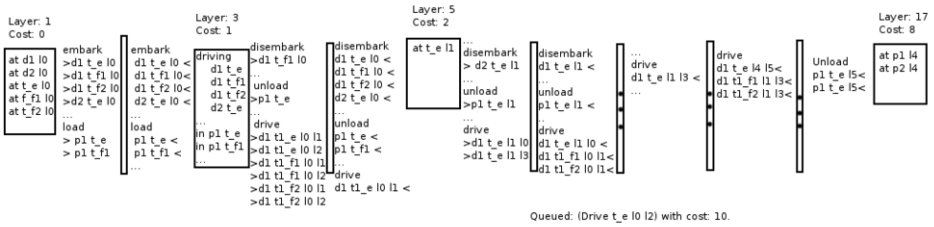


Figure 3. cost-RPG constructed from the initial state for the problem shown on Figure 1.

- Insert all facts from the initial state into Fact Layer 0.
- Then new action starts are applied, and their ends queued based on cost, until no new start action is applicable. Resulting layers are separated by ε.
- Then, lowest cost action ends are applied, this forms a new cost-RPG layer with the cost determined by the action cost.
- Then all new applicable action starts are applied.
- The process repeats until a goal is found or no new facts can be achieved.

Figure 3, demonstrates how cheap actions, in terms of the plan metrics, appear in cost-RPG and the expensive actions, in this case using fuel or driving via location 2, are postponed. It is interesting to notice that action driving to location 2 is not finished even when the goal is reached. Driving via location two is expensive and undesirable, yet the standard version of RPG uses it. In cost domains, when non-temporal action is split into start and end action, the start of an action contains no effects which can enable other actions. All positive effects, which can enable other actions, are placed at end. This ensures that the positive effects are available only after the cost is paid and, at the same time, prevents unnecessary branching.

For the purpose of handling context dependent action cost, in every state before creation of the cost-RPG, action costs are computed. These calculated action cost, within the context of the current state, are then used inside the cost-RPG for estimating the total cost of arriving to the goal state. The costs of single actions could change between different layers from cost-RPG but for simplicity we keep them constant from the time they have been calculated within the context of the state which is evaluated.

The basis of our use of POPF2 to achieve metric sensitivity lies in the way that it constructs the Temporal Relaxed Plan Graph (TRPG) [2]. The idea in the temporal setting, which is similar to the construction used in Sapa [3], is to extend the plan graph, during construction, with applicable action *start* points, queueing their end points to occur at the corresponding duration interval after the start. Once all applicable action starts have been identified and applied, arriving at a fixed point for this stage, the *earliest* queued action end point is applied and then the process is repeated, until this queue is empty (or the goals are achieved). Critically, each layer of the TRPG is labelled with the time at which it is reached. This means that the makespan of the relaxed plan can be identified directly within the TRPG and this leads to an informative temporal heuristic.

3.1. Compiling the cost-RPG

In order to exploit POPF2, without the need to modify it directly, we propose a novel compilation approach, in which non-temporal domains are compiled into 'temporal' do-

Table 1. An original action, and one of the actions resulting from a compilation.

Original Action	One of compiled temporal actions
(**:action** drive-electrictruck **:parameters** (?v - electrictruck ?f ?t - location ?d - driver) **:precondition** (and (at ?v ?f) (driving ?d ?v)(link ?f ?t)) **:effect** (and (not (at ?v ?f)) (at ?v ?t) (increase (electricity-used) (* (time-to-drive ?f ?t) (* (load ?t) (load ?t))))))	(**:durative-action** drive-electrictruck_l0-l2 **:parameters** (?v - electrictruck ?d - driver) **:duration** (= ?duration (* 100.0 (* (load ?t) (load ?t)) **:condition** (and (at start(at ?v s0)) (at start(driving ?d ?v)) (at start(link s0 s2))) **:effect**(and (at start(not (at ?v s0)))(at end (at ?v s2)) (at end (increase (electricity-used) (* 10.0 (* (load ?t) (load ?t)))))))

mains. Our method substitutes action 'durations' for action costs and, therefore, the makespan of the plan is a proxy for the plan cost. This compilation exploits the similarity between the construction of TRPG to the cost-based RPG. We now describe this process.

Faced with a non-temporal planning domain and problem instance with a single plan metric, we compile the problem into a temporal model, in which the durations are determined by the costs of the corresponding actions under the specific plan metric. The positive effects of the action are then placed at the end of the action, while the preconditions and negative effects are placed at the start.

There are several issues in the compilation. Firstly, we need to ensure that resource consumption, not to be confused with cost, occurs as soon as an action is applied, otherwise the gap between starting and ending the action can be exploited by the planner to try to execute an action that uses the same resource. Although the attempt will fail, because it will prevent the end of the original action from being applied, it will significantly impact on the search for a plan. We avoid this by ensuring that delete effects occur at the start, and add effects at the end. For numeric variables representing resources, we apply consumption effects (inhibiting) at the start and production effects (enabling) at the end. This is achieved by identifying the way in which numeric variables appear in preconditions of other actions. For example, in the Bread domain, increasing the metric fluent $(ready\text{-}dough\ ?k - kitchen)$ at the start allows actions to start baking bread before one of the $kneed\text{-}dough\text{-}machine$ or $kneed\text{-}hand$ action finishes. In this case we have to increase the $(ready\text{-}dough\ ?k)$ resource as an end effect.

It is usually necessary to translate an action from the metric domain into multiple actions in the temporal domain. One approach is to generate partially grounded versions of each action from the original domain and calculate their costs separately. In this case, we ground actions intelligently (checking static preconditions) to avoid unnecessary increase in the domain size. The compilation supports actions that do not have a constant cost and where cost depends on the context in which they are applied. Handling of this is delegated to underlying planner, which means that a planner must be able to handle context dependent action duration.

We now present an example of the compilation, consider the action presented in Table 1. For the aggregated plan metric function: $(: metricminimize(+(*10(electricity-used))(+(*1(fuel-used))(*1(walked))))$ This action is compiled into multiple durative actions. By grounding the location variables we can calculate the cost of the action given (time-to-drive ?f ?t), we ground the action for each value of (time-to-drive ?f ?t) given in the problem file. One of the translated actions is present in Table 1.

The duration of the action, 100 units multiplied by the load of the truck, is calculated using the aggregated plan metric function and grounded location functions. Knowing

that we increase electricity-used by 10 units multiplied by the load of the truck, the plan metric function increases by: $10 \times (10 \times (load?t)^2) + 1 \times 0 + 1 \times 0 = 100 \times (load?t)^2$.

4. Experiments and results

In our experiments we will show that using the cost-based approach to build the RPG can generate better quality solutions, evaluated by the plan metrics, than the closest state of the art planner. Based on our earlier experiments with the current state of the art planners, discussed in Section 2.2, LPG is selected for comparison.

Each domain defines a set of metric fluents that combined, with different weights, form a plan metric in each problem instance. For each domain and each problem instance, we generate problem instances that combine three metric fluents with weightings chosen from $\{0, ..., 10\}$ so that the sum of weights is equal to 10. this generates sixty six different weightings. Each time a planner is run it is given a 100 second window to find a solution. After 100 seconds a planner is stopped whether it finds a solution or not. This repeats for all the weighting schemes. Time of the planners reported in Section 4 is total time for generating solutions to all of the sixty six problem instances, averaged over all problems.

The main focus of this work is increasing the quality of the solutions. The method used when comparing the performance of planners with each other is based on the scoring metric used in recent International Planning Competitions. After a set of planners is run on a problem instance derived from a multi-objective problem by combining the metric fluents into a single weighted sum, the plan metric, the solutions are gathered and compared with each other. For each weighting scheme we calculate the value of the best plan across all the planners, Θ_{best} and then we assign a score to each planner, i, producing a plan for this problem with value Θ_i, using the relative quality score:

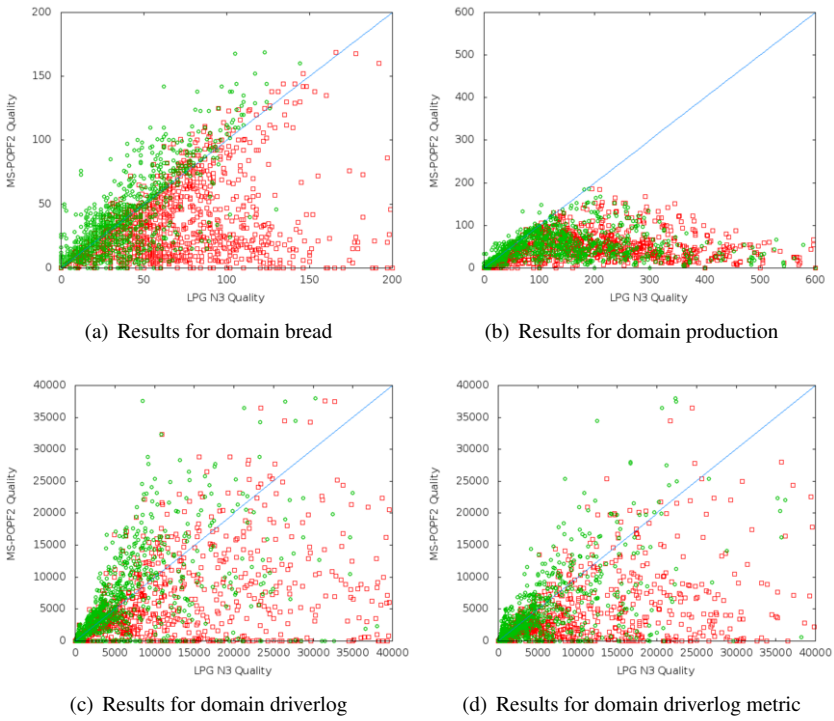
Definition 1 ***Relative Quality*** *is* $\Theta_i^{relative} = \frac{\Theta_i - \Theta_{best}}{\Theta_{best}}$
Where: Θ_i*- plan metric value;* Θ_{best}*- Best value across planners.*

This method is used to evaluate the quality of solutions for multiple runs over different plan metrics for each problem file for each domain described in Section 2.1.

Results. Experiments were conducted on an Intel®Core™i7-2600 CPU @ 3.40GHz 8 machine with 4GB or RAM memory for each planner.

Table 2 A) shows results for 13 problems from the modified Driverlog domain. It is clear that the approach based on the cost-RPG outperforms current state of the art in terms of quality. This approach takes far more time, as we depicted by average time results in Table 2 B). This can be partly explained by time contributed towards the compilation of the domain, larger amount of action in the compiled domain and the performance of POPF2 without the compilation which is also slower than LPG.

An interesting observation with regards to the role of stochasticity comes from the difference between the number of best plans found by each planner and their relative quality. This is especially visible for the Bread domain where LPG N3 generated more plans of higher quality, but its average relative quality is very low. We attribute this to its stochastic behaviour which in many cases leads to good quality solutions, but it also forces the planner to search in areas of the search space with poorer quality solutions.



(a) Results for domain bread

(b) Results for domain production

(c) Results for domain driverlog

(d) Results for domain driverlog metric

Figure 4. Results for all weights and all problems by domain aggregated together. Squares compare quality of LPG n1 and circles LPG n2 with the quality of MS-POPF2. Lower cost values are better.

5. Conclusion

We have presented a novel cost-based expansion of RPG which is a new approach to cost oriented planning. The method is analogical to temporal RPG mentioned earlier.

The comparison of performance shows that although cost-RPG requires more time to find a solution, it finds a solution of higher quality than LPG. The comparison is limited to LPG because most planner do not handle context/metric dependent action costs.

The cost-RPG heuristic works particularly well in finding plans where the number of actions does not correlate well with the change of cost. In the example presented in Section 1 there are two routes, A: $L0 - L2 - L5$ and B: $L0 - L1 - L3 - L4 - L5$. It is common for planners to favour the shorter action sequence, but more expensive, route A. In constructing the cost-RPG, the end of the action (drive-electrictruck ?v L0 L2 ?d) would not be added to the plan graph until after the cheaper route via $L1$ is already in place.

Our results, presented in Section 4, show that use of the cost-RPG leads to a metric sensitive performance that generates plans of much higher quality than the closest competitor. We have shown that LPG, as other planners, favours shorter plans over cheaper in terms of plan metrics. Our novel cost-RPG heuristic overcomes this limitation. The heuristic have been successfully implemented using a current temporal planner, POPF2, and a novel compilation from metric to temporal domain.

Table 2. A) Average relative quality results for different plan metrics for each of the problems from the modified Driverlog domain. B) Average results for relative quality, time and number of best plans generated over 20 problem files for domain Production, Bread and 13 problems for Driverlog.

	LPG1	LPG2	LPG3	MSPOPF
1	292	117	0.24	**0**
2	725	336	1.85	**0.64**
3	1149	308	136	**0.02**
4	1367	466	458	**0.25**
5	205	196	175	**0.21**
6	8	0.69	**0.2**	2.39
7	745	500	3.13	**1**
8	2915	2594	448	**0**
9	668	632	1.24	**0.61**
10	3395	1572	526	**0.32**
11	7253	5149	2956	**0.34**
12	1996	3.97	136	**0.31**
13	7272	4530	2734	**0.4**
Avg	2153	1262	583	**0.5**

Quality	LPG1	LPG2	LPG3	MSPOPF
Production	12.52	7.34	6	**0.29**
Bread	8.94	2.75	1.32	**0.55**
Driver	2024	1059	443	**0.65**
Driver_m	1699	688	257	**2.11**
Time	LPG1	LPG2	LPG3	MSPOPF
Production	**6.23**	56.91	305.5	602
Bread	**5.14**	151.37	1254.71	22.6
Driver	**8.09**	13.88	64.27	262.47
Driver_m	**8.92**	13.92	63.92	439.5
# Best	LPG1	LPG2	LPG3	MSPOPF
Production	33	220	401	**966**
Bread	35	323	**707**	584
Driver	99	302	**554**	413
Driver_m	47	186	358	**436**
Total	214	1031	2020	**2399**

References

[1] A. Coles, M. Fox, D. Long, and A. Smith, 'A hybrid relaxed planning graphlp heuristic for numeric planning domains', *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, (2008).

[2] A. J. Coles, A. I. Coles, M. Fox, and D. Long, 'Forward-chaining partial-order planning', in *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, (2010).

[3] Minh Binh Do and S. Kambhampati, 'Sapa: A multi-objective metric temporal planner', *J. Artif. Intell. Res.*, **20**, 155–194, (2003).

[4] M. Fox and D. Long, 'PDDL 2.1: An Extension to PDDL for Expressing Temporal Planning Domains', *J. Artif. Int. Res.*, **20**, 61–124, (2003).

[5] A. Gerevini, A. Saetti, and I. Serina, 'LPG-TD: a Fully Automated Planner for PDDL2.2 Domains', *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, (2004).

[6] J. Hoffmann and B. Nebel, 'The FF Planning System: Fast Plan Generation Through Heuristic Search', *J. Artif. Int. Res.*, **14**, 253–302, (2001).

[7] Emil Keyder and Hector Geffner, '"heuristics for planning with action costs revisited"', in *ECAI*, pp. 588–592, (2008).

[8] D. Long and M. Fox, 'The 3rd International Planning Competition: Results and analysis', *J. Artif. Int. Res.*, **20**, 1–59, (2003).

[9] Nor Haizan Mohamed Radzi, *Multi-Objective Planning using Linear Programming*, Ph.D. dissertation, University of Strathclyde, 2011.

[10] S. Richter and M. Westphal, 'The LAMA Planner: Guiding Cost-based Anytime Planning with Landmarks', *J. Artif. Int. Res.*, **39**, 127–177, (2010).

[11] M. Sroka and D. Long, 'Exploring Metric Sensitivity of Planners for Generation of Pareto Frontiers.', in *Starting AI Research Symposium (STAIRS)*, pp. 306–317, (2012).

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-280

Towards Learning and Classifying Spatio-Temporal Activities in a Stream Processing Framework

Mattias TIGER [a] and Fredrik HEINTZ [a]
[a] *Department of Computer and Information Science, Linköping University, Sweden* [1]

Abstract. We propose an unsupervised stream processing framework that learns a Bayesian representation of observed spatio-temporal activities and their causal relations. The dynamics of the activities are modeled using sparse Gaussian processes and their causal relations using a causal Bayesian graph. This allows the model to be efficient through compactness and sparsity in the causal graph, and to provide probabilities at any level of abstraction for activities or chains of activities. Methods and ideas from a wide range of previous work are combined and interact to provide a uniform way to tackle a variety of common problems related to learning, classifying and predicting activities. We discuss how to use this framework to perform prediction of future activities and to generate events.

Keywords. activity recognition, machine learning, knowledge representation, situation awareness, knowledge acquisition, unsupervised learning

Introduction

Learning and recognizing spatio-temporal activities from streams of observations is a central problem in artificial intelligence. Activity recognition is important for many applications including detecting human activities such that a person has forgotten to take her medicine in an assisted home [1], monitoring telecommunication networks [2], and recognizing illegal or dangerous traffic behavior [3,6]. In each of the applications the input is a potentially large number of streams of observations coming from sensors and other information sources. Based on these observations common patterns should be learned and later recognized, in real-time as new information becomes available.

The main contribution of this paper is an unsupervised framework for learning activities and causal relations between activities from observed state trajectories. Given a set of state space trajectories, the proposed framework segments the continuous trajectories into discrete activities and learns the statistical relations between activities as well as the continuous dynamics of each activity. The dynamics of the activities are modeled using sparse Gaussian Processes and their contextual relations using a causal Bayesian graph. The learned model is sparse and compact and supports both estimating the most likely activity of an observed object, and predicting the most likely future activities.

[1] Corresponding Authors: {Mattias Tiger, Fredrik Heintz}, Department of Computer and Information Science, Linköping University, Sweden; E-mail: {matti166@student.liu.se, fredrik.heintz@liu.se}.

As an example, consider a traffic monitoring application where a UAV is given the task of monitoring the traffic in a T-crossing. The UAV is equipped with a color and a thermal camera and is capable of detecting and tracking vehicles driving through the intersection [6]. The output of the vision-based tracking functionality consists of streams of states where a state represents the information about a single car at a single time-point and a stream represents the trajectory of a particular car over time (Figure 1, left). Based on this information the UAV needs to learn traffic activities such as a car driving straight through the intersection or making a turn (Figure 1, right).

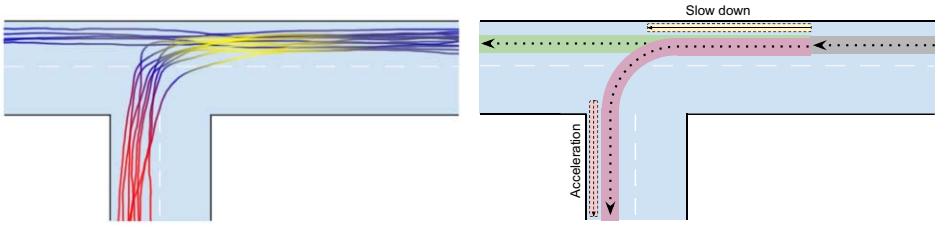


Figure 1. An illustrative example of the expected input and output of the proposed framework. The left figure shows observed trajectories. Blue is constant velocity, yellow is negative acceleration, and red is positive acceleration. The right figure shows a conceptual illustration of what we want to learn.

1. Related Work

There are many different approaches to activity recognition, both qualitative and quantitative. Here we focus on quantitative approaches since they are most related to our work. As far as we know there is no other unsupervised framework that can learn the atomic activities, the dynamics of each activity and the causal relations between the activities.

A common approach is to use images as the raw input. One approach uses a Bayesian Hierarchical Model with moving pixels from video surveillance as input [17]. They divide the image into square regions which the pixels belong to and use together with four different motion directions as a code book. They treat 10 second video clips as documents and the moving pixels as words and learn a Hierarchical Dirichlet Process. By performing word document analysis, the resulting topics are normative atomic activities in the form of pixel motion trajectories in the 2D image.

Our unsupervised framework in comparison requires trajectories, but can learn primitive activities bottom up from these trajectories. This allow our framework to rapidly learn new activities, while being able to compare their support by each activity's accumulated evidence. We also learn the causal statistical connections between activities allowing for activity prediction. Our framework could use moving pixels as input, but also other types of information such as 3D-positions. It also incorporates the uncertainties of those measurements.

Another approach is to use splines with rectangular envelopes to model trajectories [10,5], compared to our approach and the approach of Kim [8] which use Gaussian Processes (GPs) in a continuous Bayesian setting. Similar to our approach [5] divides trajectories into shorter segments and split trajectories at intersections, while [8] only

considers trajectories beginning and ending out of sight and performs no segmentation of activities. Neither of the frameworks mentioned are suitable in a stream processing context with incrementally available information in the learning phase. Our framework can continue to learn in the field on a robotic platform and adapt directly to new situations in regards to learning, classification and prediction. Our framework supports not only detecting and keeping count of abnormal behaviors, but also learns those behaviors and collect statistics on their detailed activities.

2. The Activity Learning and Classification Framework

The proposed framework (Figure 2) learns activities and their relations based on streams of temporally ordered time-stamped probabilistic states for individual objects in an unsupervised manner. Using the learned activities, the framework classifies the current trajectory of an object to belong to the most likely chain of activities. The framework can also predict how abnormal the current trajectory is and how likely future activities are.

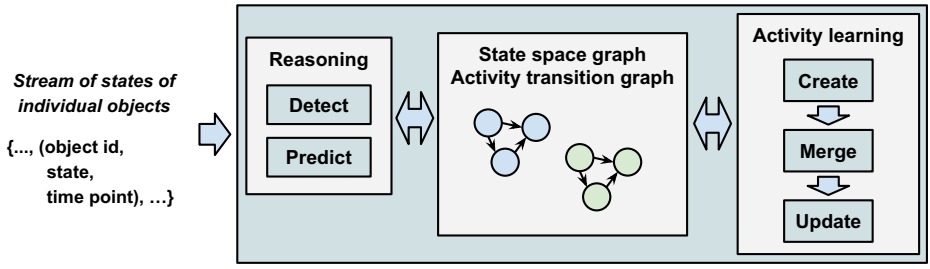
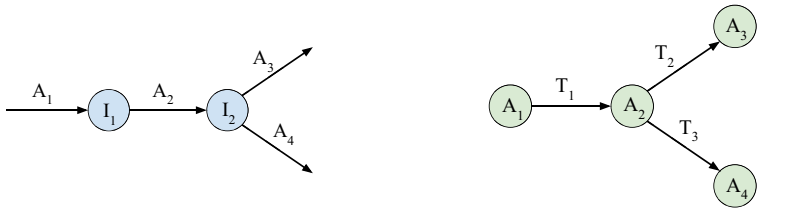


Figure 2. An overview of the framework. It is built around the three modules *Reasoning*, *Activity Learning* and the knowledge base consisting of two graphs, of which the two former modules make extensive use of. The *Activity learning* takes entire observed trajectories and updates the graphs. It is divided into three parts which are performed sequentially. The *Reasoning* module detects and predicts activities based on observations.

An activity can informally be defined as something a particular object does. In our case, an instance of an activity is represented by a state trajectory where some state variables are expected to change in a certain way while others may change freely. For example, an activity could be *slow down* where the velocity state variable is expected to decrease. An activity model should capture such continuous developments.

The framework models activities as Gaussian Processes, which are sufficiently expressive to model the development of the state variables including uncertainties in observations. Activities are learned as edges in a graph, where the nodes are intersection points, in the state-space, between activities. On a higher abstraction level activities are seen as nodes, with edges representing transitions which are weighted according to the observed empirical probability of the transitions. The latter construct enables prediction of activities through probabilistic inference. The graphs are called the *State Space Graph* and the *Activity Transition Graph*. An example is shown in Figure 3. The Activity Transition Graph is a causally ordered discrete Markov chain. We currently only consider 1-order Markovian transitions in the formulation of this graph. However, a natural extension is to use a higher order Markov chain.



(a) State Space Graph (b) Activity Transition Graph

Figure 3. *A* denotes an activity in both the State Space Graph and the Activity Transition Graph, *I* denotes an intersection point in the state space, *T* denotes a transition in the Activity Transition Graph. All transitions between A_2, A_3 and A_4 can be seen as contained within the intersection point I_2.

To handle activities on different scales the framework uses two parameters $s_{threshold}$ and $l_{threshold}$. The parameter $s_{threshold}$ is a threshold on the similarity between activities in the state space, based on the likelihood of one explaining the other. If an observed trajectory has a similarity below the threshold when compared to any other activity, the observation is considered novel and unexplained. The parameter $l_{threshold}$ is a threshold on the minimal length of an activity in the state space, which is used to limit the amount of detail in the learned model.

The framework captures and models activities as statistically significant trajectories through the state space. With enough observations all activities captured are discriminative and invariant of the observed variations which are not statistically significant.

In the work of [8] they detect abnormal trajectories by using the lack of dominant trajectory models in explaining observed trajectories. In our case a similar lack of dominance of activity models, or that several are almost equally likely, can be due to two things. Either due to abnormality (novelty for learning) or due to a transition between activities. By using the similarity threshold $s_{threshold}$ we can distinguish between intervals that are either abnormal or transitional.

3. Modeling Activities

An activity is seen as a continuous trajectory in the state space, i.e. a mapping from time t to state y. A state can for example be a 2D position and a 2D velocity. To get invariance in time, t is normalized to be between 0.0 and 1.0 and acts as a parametrization variable of $y = f(t)$. The mapping is modeled using Gaussian Processes. The assumption is that an activity is a curve in a N-dimensional state space. The activity model is the result of the continued accumulation of explained observed trajectories. The amount of these supporting trajectories is seen as the amount of evidence E of the activity model, and is used in comparison with other activity models.

Given an observed state, this parametrization variable must be known to calculate how well the activity explains the observation. In [8] they assume this parametrization to be known by assuming that it is already normalized to the interval $[0,1]$ which can only occur after the whole trajectory has been observed. Further, they only consider entire trajectories without temporal segmentation, and with a state space consisting of positions and velocities. When performing stream reasoning the entire trajectory cannot be assumed to be present at once such that it can be normalized, because states are made

available incrementally. We estimate the parametrization for any observation to allow the system to operate with only incrementally available pieces of the observed trajectories.

To estimate the temporal parameter for any observed state, an inverse mapping $t = g(y)$ is also learned. This adds a bijective restriction on an activity, meaning that an activity cannot intersect itself. Concretely, it cannot map to the same state at different points in time. This is a limitation arising from the non-causality of Gaussian Processes used to model the activities. The limitation can be circumvented by segmenting observed state trajectories into non-intersecting intervals and model each separately.

3.1. Gaussian Processes

Gaussian Processes have been shown to be useful for prediction, modeling and detecting spatio-temporal trajectories such as motor vehicles in crossings [8], marine vessel paths [12] and human body dynamics [16]. GPs have also been used to model spatial regions and structures such as occupancy maps [9] and smooth motion paths [7]. They are also good for handling noisy or missing data [8,16], where large chunks of trajectories can reliably be reconstructed.

One major obstacle restricting their use has in the past been the expensive matrix inversion operation necessary to update the model. Various methods have been employed to reduce this weakness, by efficient iterative updates [13], segmentation into local patches of GPs [11,15], and by techniques to make the GP sparse [14,15]. In this work we employ sparse GPs to model activities, which are systematically segmented into local models.

The non-parametric model of the Gaussian Process is associated with a vector of inputs $\mathbf{x}$ and two corresponding vectors of outputs $\mathbf{y}$ and output uncertainties $\sigma_{\mathbf{y}}$. The posterior density given an observed point x^* is a Gaussian distribution. We use a sparse representation of support points $\mathbf{x}$ and $\mathbf{y}$, kept to a limited number, to make the calculations possible with any number of accumulated observations. The intention is to make it possible for the system to continue to learn from as many observations as possible to improve the confidence and get as much evidential support as possible.

3.2. Modeling Spatio-Temporal Dynamics

The activity model consist of the parametrized state trajectory $y = f_{GP}(t)$ and its inverse mapping $t = g_{GP}(y) = f_{GP}^{-1}(y)$. The inverse mapping is constructed by switching the input and the output, $\mathbf{x}$ and $\mathbf{y}$, of f_{GP}. Since the inverse mapping is with respect to the mean function of f_{GP}, it is exact at these points and the observation error is therefore zero. It is however chosen to be very small as to minimize numerical problems.

The main purpose of the inverse mapping is to project a state space point $x^* \sim \mathcal{N}(\mu_{x^*}, \sigma_{x^*}^2)$ onto the mean function of f_{GP}, thereby becoming the state space point $y^* \sim \mathcal{N}(\mu_{y^*}, \sigma_{y^*}^2)$. This is used when calculating the likelihood of a point on one trajectory being explained by a point on another. The calculation is performed by approximating x^* as μ_{x^*} and t^* as μ_{t^*},

$$y^* = f_{GP}(\mu_{t^*}), \quad t^* = g_{GP}(\mu_{x^*}). \tag{1}$$

The latter approximation is valid if the inverse mapping accurately models the inverse mean function. The former approximation can be improved by techniques for uncertain

inputs to GPs described in [4]. The mappings $y = f_{GP}(t)$ and $t = g_{GP}(y)$ currently use the squared exponential kernel, which is commonly used in Gaussian Process regression,

$$k(x_1, x_2) = \theta_0^2 e^{-\frac{||x_1 - x_2||_2^2}{2\theta_1^2}}, \tag{2}$$

where θ_0^2 denotes the global variance of the mapping and θ_1^2 denotes the global smoothness parameter of the mapping. The two mappings have separate parameters. The kernel hyper parameters θ are optimized for each activity by maximizing the marginal likelihood.

3.3. Local Likelihood

Let A be an activity with $\{f_{GP}, g_{GP}\}$ and let $x^* \sim \mathcal{N}(\mu_{x^*}, \sigma_{x^*}^2)$ be an observed point or a point on another activity. If the different dimensions of the state space are assumed to be independent, the local likelihood[8] of x^* given A is

$$P(x^*|A) = P(x^*|y^*) = \prod_{n=0}^{N} P(x_n^*|y_n^*), \qquad y^* = f_{GP}(g_{GP}(x^*)), \tag{3}$$

where N is the dimension of the state space and y^* is a Gaussian distribution. When considering multiple activities the calculation of the local likelihood is weighted by the relative evidence E in support of respective activity. If T_n is point n on trajectory T then

$$P^*(T_n|A_k) = \frac{E_k}{\sum_i (E_i)} P(T_n|A_k), \tag{4}$$

where i ranges over all activities in consideration of which A_k is one. This allows a well supported activity with high variance to keep significant importance.

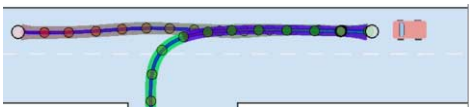
4. Learning Activities

To learn activities from observed trajectories, the framework segments trajectories into sequences of activities, where each activity differs from all other activities currently known. Activities are segmented such that no two activities can have overlapping intervals in the state space, where the margin for overlap is controlled by $s_{threshold}$.

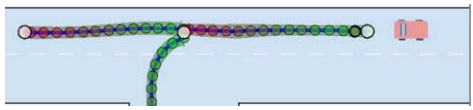
The learning process updates the State Space Graph and the Activity Transition Graph based on an observed trajectory. The first step is to generate a GP from the observed trajectory, and then to generate a new trajectory by uniformly sample the GP to get a smaller number of support points for computational efficiency. The next step is to classify the trajectory given the current set of activity models. Based on this classification the graphs are updated. If a long enough interval of an observed trajectory lacks sufficient explanation given the known activities, then this interval is considered novel and is added as a new activity. Intervals that are explained sufficiently by a single activity more than by any other is used to update that activity with the new observed information. Sometimes long enough parts of activities become too similar, i.e. too close together in the state space, to be considered different activities. In those cases such parts of these ac-

tivities are merged. This removes cases where intervals of the observation is sufficiently and on equal terms explained by more than one activity.

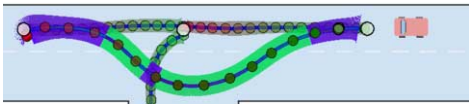
By letting the transition nodes store statistics of the transitions between the activities, the Activity Transition Graph becomes another perspective of the same information contained in the State Space Graph. It is updated simultaneously with the State Space Graph in the different learning steps.



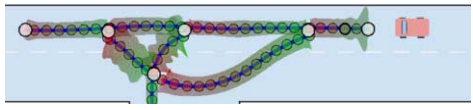
(a) An activity of driving straight and an observation of a left turn. Bright blue indicates interval for update. Bright green indicates a interval which will result in the creation of a new activity.



(b) Now three activities after learning the left turn. The previous activity is now split at the point where the left turn was no longer explained sufficiently by the straight activity.



(c) An illegal overtake is observed. The rounded trajectory of the overtake is indicated to be created as two activities since it is too similar to the left turn when spatially crossing it. This do not happen if velocities are part of the state space.



(d) After the overtake has been learned. The highly varying variance seen on the shorter activities is due to the parameters of the kernels not being updated properly.

Figure 4. Example of the learning of the State Space Graph using only 2D positions as state variables for the activities, in order (a)-(d). Activity coloring is from green (t = 0.0) to red (t = 1.0). The white circles are intersection points, the other circles are support points of the activities' f_{GP}.

4.1. Learning New Activities

A new activity is created for each maximal interval $\tau := [t_0 : t_1]$ where $P(T_\tau|A_k) < s_{threshold}$ for all activities A_k in consideration and where $\Delta l_\tau > l_{threshold}$. That is, any long enough interval of the observed trajectory, which is insufficiently explained by every concerned activity, is added to the framework as a new activity.

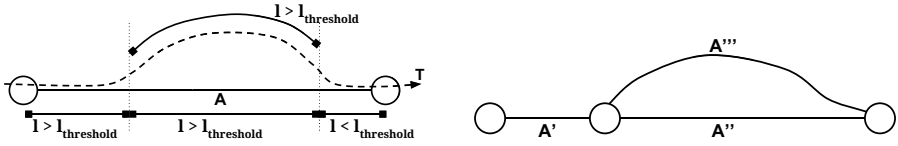


Figure 5. Illustrative example of the *create procedure*, where the left figure is before and the right figure is after. The dotted lines indicate where the similarity of A and T is crossing the $s_{threshold}$.

Intersection points are used to connect activities with other activities. Since activities are not allowed to be shorter than $l_{threshold}$, it is also the case that two intersection points cannot be created on the same activity with a distance between them shorter than $l_{threshold}$.

A new intersection point is created at each end if the length to the nearest other intersection point is larger or equal to $l_{threshold}$. If not, the new activity is connected to the nearest intersection point and the intersection point is updated to be placed at the weighted mean position of the end-points of the connected activities. The weighting is done in accordance to the amount of evidence of each connected activity.

If a new activity is branching into or out of a previous activity, the previous activity is split in two at the branching point. The intersection point is placed at the weighted mean position.

If an intersection point after its placement update is closer than $l_{threshold}$ to another intersection point, they are merged into a single intersection point. An illustrative example is shown in Figure 5.

4.2. Merging Activities

To remove redundant activities, activities are merged at intervals that are too similar to each other. They are also merged in order to allow for spatially wide activities, which might be initially observed as several parallel activities growing together over time as more observations are acquired.

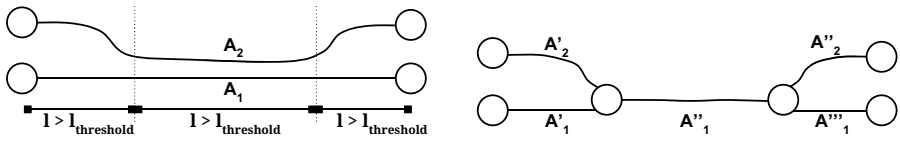


Figure 6. An illustrative example of the *merge procedure*, where the left figure is before and the right figure is after. The dotted lines indicate where the similarity of both A_2 and A_1 are exceeding the $s_{threshold}$

All activities A_k are merged with A^* on the interval $\tau := [t_0 : t_1]$ for which $P(T_\tau|A_k) \geq s_{threshold}$, $\Delta l_\tau > l_{threshold}$ and A^* is the activity with the maximum similarity $P(T_\tau|A^*)$. That is, all activities which sufficiently explain long enough intervals of the observed trajectory are merged with the activity best explaining that interval.

Activities are currently merged and updated by updating the most supported activity's set of GP support points within the given interval with the corresponding state space points sampled from the other activities. The new state space location of the support point is the weighted mean of its previous location and the sampled location. The weighting is based on the number of previous observations in support of respectively activity, used as a measure of evidence for respective activity model. The final activity has the combined amount of evidence from all activities merged. Intersection points are created and updated in the same way as before. An illustrative example is shown in Figure 6.

4.3. Updating Activities

Any interval τ_i on T not used for creating or merging in the two previous steps carry information that are used to update activities. The activity best explaining such an interval cannot have been created or been part of a merge in the previous two steps.

The intervals remaining with $length > l_{threshold}$, $I_n, n = 1...N$, always have at every point an activity A^* with maximum similarity where $P(T_t|A^*) \geq s_{threshold}$ holds except for intervals with $length \leq l_{threshold}$. These intervals are segmented into shorter intervals

such that there is only a single activity with maximum similarity for the entire interval, $I_{n,i}$.

If $length(I_{n,i}) \leq l_{threshold}$ it is combined with the neighbor of shortest length until they are composite intervals satisfying $length(Ic_{n,i}) > l_{threshold}$. The activity assigned as the most likely to the composite interval is the activity A_k which has the highest $P(Ic_{n,i}|A_k)$ for all k in consideration.

The intervals and composite intervals are merged with the most similar activity as described in the previous section, with the weight of the intervals equal to one since they only represent a single observation.

5. Detecting and Predicting Activities

The State Space Graph and the Activity Transition Graph can be used in several ways to detect and predict activities. There is statistical information such as the evidence for each activity, similarity between any two activities and the probability of transition between activities. Assuming the presence of one or several learned activities, it is straightforward to calculate the activity chain from n steps in the past to m steps into the future that is most likely, least likely or anything in between. It is also possible to calculate the k most likely activity chains, with or without sharing activities at any step.

Another possibility is to generate relevant events as part of the learning procedure. Events can for example be created when the learning procedure creates, updates and merges activities, or when an object has completed an activity due to the object crossing an intersection point.

6. Preliminary Results

To verify that our approach works as expected, we have done a case study with a T-crossing scenario. The data set contains two types of trajectories, straight trajectories and left turning trajectories, 50 of each. These trajectories are provided to the framework in a random order. The converged trajectories can be seen in Figure 7 compared to the generated data. With position and constant velocity the turning activity breaks up a bit earlier than with only position, and even more so with the slow down taking place.

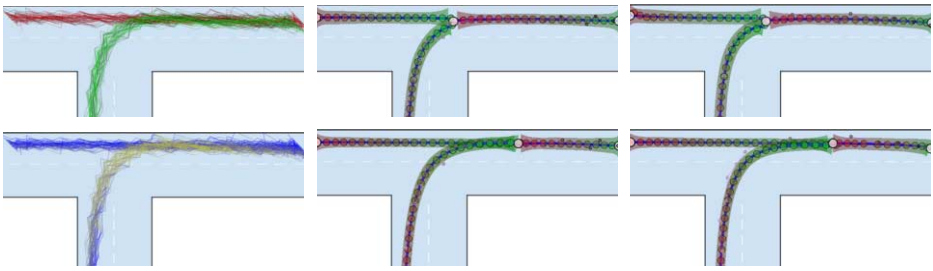


Figure 7. Case study. Top row: Scenario with positions only. Bottom row: Scenario with a slow down. The figures in the middle show the three learned activities after two observed trajectories, while the figures to the right show the result after all observations. Blue color indicate a higher speed than yellow.

7. Conclusions and Future Work

Learning and recognizing spatio-temporal activities from streams of observations is a central problem in artificial intelligence. In this paper, we have proposed an initial approach towards an unsupervised stream processing framework that learns a Bayesian representation of observed spatio-temporal activities and their causal relations from observed trajectories of objects. An activity is represented by a Gaussian Process and the set of activities is represented by a State-Space Graph where the edges are activities and the nodes are intersection points between activities. To reason about chains of related activities an Activity Transition Graph is also learned which represents the statistical information about transitions between activities. To show that the framework works as intended, it has been applied to a small traffic monitoring example.

The framework cannot yet learn complex activities as discrete entities other than indirectly by parts, i.e. by learning chains of primitive activities. Neither are contexts such as *a T-crossing* learned, instead configurations of different activities are learned for each T-crossing instance. The current framework is designed to provide the building blocks for these extensions to be possible in future work.

References

[1] J. Aggarwal and M. Ryoo. Human activity analysis: A review. *ACM Comput. Surv.*, 43(3), 2011.

[2] C. Dousson and P. Le Maigat. Chronicle recognition improvement using temporal focusing and hierarchization. In *Proc. IJCAI*, 2007.

[3] R. Gerber and H.-H. Nagel. Representation of occurrences for road vehicle traffic. *Artificial Intelligence*, 172(4–5):351–391, 2008.

[4] A. Girard, C. Rasmussen, J. Candela, and R. Murray-Smith. Gaussian process priors with uncertain inputs - application to multiple-step ahead time series forecasting. In *Proc. NIPS*, 2002.

[5] N. Guillarme and X. Lerouvreur. Unsupervised extraction of knowledge from S-AIS data for maritime situational awareness. In *Proc. FUSION*, 2013.

[6] F. Heintz, P. Rudol, and P. Doherty. From images to traffic behavior - a uav tracking and monitoring application. In *Proc. FUSION*, 2007.

[7] C. Tay Meng Keat and C. Laugier. Modelling smooth paths using gaussian processes. In *Proc. FSR*, 2007.

[8] K. Kim, D. Lee, and I. Essa. Gaussian process regression flow for analysis of motion trajectories. In *Proc. ICCV*, 2011.

[9] S. Kim and J. Kim. Continuous occupancy maps using overlapping local gaussian processes. In *Proc. IROS*, 2013.

[10] D. Makris and T. Ellis. Learning semantic scene models from observing activity in visual surveillance. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(3):397–408, 2005.

[11] M. Schneider and W. Ertel. Robot learning by demonstration with local gaussian process regression. In *Proc. IROS*, 2010.

[12] M. Smith, S. Reece, I. Rezek, I. Psorakis, and S. Roberts. Maritime abnormality detection using gaussian processes. *Knowledge and Information Systems*, pages 1–26, 2013.

[13] M. Smith, S. Reece, S. Roberts, and I. Rezek. Online maritime abnormality detection using gaussian processes and extreme value theory. In *Proc. ICDM*, 2012.

[14] E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Proc. NIPS*, 2005.

[15] E. Snelson and Z. Ghahramani. Local and global sparse gaussian process approximations. In *Proc. AISTATS*, 2007.

[16] J. Wang, D. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(2):283–298, 2008.

[17] X. Wang, X. Ma, and E. Grimson. Unsupervised activity perception by hierarchical bayesian models. In *Proc. CVPR*, 2007.

STAIRS 2014
U. Endriss and J. Leite (Eds.)

doi:10.3233/978-1-61499-421-3-290

Empirical Study of Classification Models for Web Page Categorization

Tomáš TUNYS [a], Jan ŠEDIVÝ [b]
[a] *Czech Technical University in Prague, tunystom@fel.cvut.cz*
[b] *Czech Technical University in Prague, sedivja2@fel.cvut.cz*

Abstract. We describe one part of a web page content classification system used for contextual advertising. The system classifies the content of a web page to one of many predefined ad categories, i.e. performing text classification [2,8,10]. Our goal is to identify the best performing classification model for Czech language. We present a comprehensive comparison study of selected models, text representations, and feature selection techniques on a collection of datasets with different numbers of documents, numbers of categories, and varying category sizes. We conclude the work with the recommendation for the best performing models.

Keywords. text classification, text representation, feature selection, model comparison, naive bayes model, dirichlet compound multinomial

Introduction

Text classification is a frequently studied problem mostly with documents in English. Our task was to find a good solution for Czech, which is morphologically more complex language, and despite the fact that we believe there is no strong evidence that best models for English should not work as well for Czech, we wanted to prove this assumption. We approached the problem by comparing currently successful models for text classification in English with models for Czech.

For comparison purposes we have selected publicly available datasets of different sizes and different number of categories. For validity of our results across foreign-language corpora, we paid a lot of attention to select publicly available corpora, which are comparable to our Czech datasets in the size, number of classes, and content (web pages). The chosen datasets are described in detail in Section 2.

The collection of studied models starts with a simple model such as Multinomial Naive Bayes model together with its advanced versions referred to as Transformed Weight-normalized Complement Multinomial Naive Bayes models [7], continuing with more elaborate generative models such as Dirichlet Compound Multinomial model [4]. For completeness we included the discriminative SVM model into our experiments, which is, to our knowledge, currently the best performing classification model. The selected models are described in Section 1.

Since Czech is a morphologically rich language, i.e. its words take many different forms, we decided to inspect the impact of different feature selection techniques and vocabulary reduction methods together with lemmatization on the classification perfor-

mance of the models. More details about the considered preprocessing techniques are in Section 3 and 5.

The combination of datasets, feature selection techniques, models and different parameters led to a vast number of experiments, which consumed a large number of computations resulting in even larger number of results. For the lack of space we limited the size of the reported results. Nevertheless, we show and discuss the main outcomes, with a small possibility of omitting a few details here and there.

Finally, we provide the results of the found optimal combinations of parameters and preprocessing steps over models and datasets. We conclude with providing recommendations that turns out to be valid across the different corpora, languages, and models.

1. Classification Models

This section provides a brief description of classification models and various transformations of a the simple vector space representation of documents.

1.1. Multinomial Naive Bayes Model

Multinomial Naive Bayes (MNB) is one of the most common text classification models. The model assigns probability to a document represented as a vector of word counts $\boldsymbol{x} = (x_1, x_2, \ldots, x_\mathcal{V})$ under the document category c according to

$$p(\boldsymbol{x}|\boldsymbol{\theta}_c) = \frac{\left(\sum_{w=1}^{\mathcal{V}} x_w\right)!}{\prod_{w=1}^{\mathcal{V}} x_w!} \prod_{w=1}^{\mathcal{V}} \theta_{cw}^{x_w} \tag{1}$$

where $\mathcal{V}$ is the size of the vocabulary and $\boldsymbol{\theta}_c$ are the word emission probabilities which can be estimated using maximum likelihood [7]. A test document $\boldsymbol{x}$ is assigned to the category c with the highest probability

$$p(c|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{\theta}_c)p(c)}{\sum_{c=1}^{\mathcal{C}} p(\boldsymbol{x}|\boldsymbol{\theta}_c)p(c)} \tag{2}$$

where the *a priori* probability $p(c)$ are commonly estimated from the training set using maximum likelihood.

To fight against the tendency of MNB to overestimate emission probabilities because of the violation of the independence assumption, a non-Bayesian alternative of the MNB model called *(Transformed) Weighted-normalized Multinomial Bayes* [7] was devised. We used both these models in our experiments to assess the relative improvement of the latter over the former.

1.2. Dirichlet Compound Multinomial Model

Dirichlet Compound Multinomial Model (DCM) is a two-level hierarchical Bayesian model that can be viewed as an extension of the MNB model and as such, it can be understood as bag-of-bag-of-words [4]. The advantage of DCM model over MNB model is that it accounts for word *burstiness*. The word burstiness refers to phenomenon which

describes the natural tendency of words to appear in documents multiple times. The a priori probability of a word appearing in a document may be quite low, but once the word appears its probability of appearing again is much higher (less surprising). The strong independence assumptions of MNB model can never capture this behaviour.

The model assigns probability to a document represented as a vector of word counts $\boldsymbol{x} = (x_1, x_2, \ldots, x_{\mathcal{V}})$ under the category c according to

$$p(\boldsymbol{x}|\boldsymbol{\alpha}_c) = \frac{\left(\sum_{w=1}^{\mathcal{V}} x_w\right)!}{\prod_{w=1}^{\mathcal{V}} x_w!} \frac{\Gamma(\sum_{w=1}^{\mathcal{V}} \alpha_w)}{\Gamma(\sum_{w=1}^{\mathcal{V}} x_w + \alpha_w)} \prod_{w=1}^{\mathcal{V}} \frac{\Gamma(x_w + \alpha_w)}{\Gamma(\alpha_w)} \tag{3}$$

where $\mathcal{V}$ is the vocabulary size and Γ is the gamma function (see [4] for the derivation). It is obvious on the first glance that calculating the probability of a document under DCM model (Eq. 3) is much more complicated in comparison with MNB model (Eq. 1).

Sadly, no closed-form solution for the maximum likelihood estimate of the $\boldsymbol{\alpha}_c$ exists, but there are plenty of iterative gradient ascent optimization methods [6], from which the method we used to train the models in our experiments is a fixed-point iteration.

The classification of a test document is done similarly as in case of the MNB model using the class membership probability, see Eq. (2).

1.3. Complementary MNB and DCM Models

Complementary modeling with MNB was introduced to deal with corpora that contain skewed document classes [7], and it showed promising results for DCM models as well [4]. We refer to the complement alternatives of the two models as CNB and CDCM.

In regular versions of MNB and DCM models the parameters are estimated from documents of particular category c, on the contrary, the CNB and CDCM models estimate the parameters from the documents which belong to all the categories *except c*. For the explicit formulas on parameter estimation and prediction see [7,4].

1.4. Text Document Representations

To improve the classification performance of a multinomial models several heuristics based on transformation of word count vectors have been proposed [7]. We have implemented 15 different transformations which altogether with the original vector make 16 different document vector representations.

Consider a corpora containing $\mathcal{D}$ documents, represented as count vectors $\boldsymbol{x}_d = (x_{d1}, x_{d2}, \ldots, x_{d\mathcal{V}})$, we define the transformations as follows

$$x^1_{dw} = x_{dw}$$

$$x^2_{dw} = \frac{x_{dw}}{\max'_w x_{dw'}} \qquad x^{i+4}_{dw} = x^i_{dw} \log \frac{\mathcal{D}}{\sum_{d=1}^{\mathcal{D}} \delta_{dw}}, \quad i = 1, \ldots, 4$$

$$x^3_{dw} = \log(1 + x^1_{dw}) \qquad x^{i+8}_{dw} = \frac{x^i_{dw}}{\sqrt{\sum_{w=1}^{\mathcal{V}} (x^i_{dw})^2}}, \quad i = 1, \ldots, 8$$

$$x^4_{dw} = \log(1 + x^2_{dw})$$

where δ_{dw} is 1 iff word w occurs in document d. Note that the document representations $x^5, \ldots, x^{16}$ are referred to as *tf-idf* counts in Information Retrieval community.

2. Datasets

To assess the quality of the models across languages we used our Czech corpora together with commonly used corpora for text classification in English. There is to our knowledge no evidence that the performance of any of the considered model is language dependent, but to make sure we can spot this (theoretical) dependency, we selected the English datasets carefully to resemble our Czech datasets in the number of categories, content, and the size.

2.1. English Datasets

The 4 Universities dataset[1] (denoted as `webkb`) contains web pages collected at computer science departments of four universities. The 8,282 pages were manually classified into the 7 categories from which we used only 4 similarly to [3], namely `student`, `faculty`, `course`, and `project`, resulting in 4200 documents.

The 20 Newsgroups[2] (denoted as `20news`) is a popular benchmarking collection for document classification models. It consists of approximately 20k newsgroup documents, partitioned evenly across 20 different categories.

The industry sector dataset[3] (denoted as `sector`) is another popular document collection for benchmarking of text classification algorithms. The data set comprises of 95470 corporate web pages partitioned into 104 categories.

2.2. Czech Datasets

We have 2 collections of Czech web pages. The web pages from both of these collections are divided into 36 categories. We received these datasets from Czech company Seznam.cz, the 36 categories correspond to web page categories used in their Sklik.cz pay per click system. We refer to these collection as `seznam1k` and `seznam11k` where the former contains 1818 documents and the latter contains 11114 documents.

3. Dataset Preprocessing

This section summarizes the basic preprocessing steps, which were applied to datasets prior to the application of feature selection and subsequent training of the models. Note that before any of these techniques is applied, the datasets are rid of HTML tags, stopwords, numbers, special characters, and whole text is put to lowercase.

[1] Available at: `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/webkb-data.gtar.gz`

[2] Available at: `http://qwone.com/~jason/20Newsgroups/20news-18828.tar.gz`

[3] Available at: `http://people.cs.umass.edu/~mccallum/data/sector.tar.gz`

3.1. Vocabulary Pruning

A simple technique relying on the assumption that very rare and very common words are not likely to be relevant because among the former, technical terms and typos prevail in terms of word occurrences, on the other hand in the latter, words like "the", "of", "and" are the most common (in English) which are not very useful terms for classification.

Final step in preprocessing of text documents is to count the document frequency of each word, i.e. in how many documents the word appears. If the frequency is below a chosen absolute (lower) threshold (e.g. 10 documents) or more then a chosen relative (upper) threshold (e.g. 50% of documents) the word is discarded. The words that are shorter than 3 or longer than 20 characters are being discarded as well.

3.2. Lemmatization

Our target language is Czech, which is morphologically more complex than English. It is common for many Czech words (especially for nouns, verbs, and adjectives) to occur in many distinct forms, so-called inflections, therefore vector representations of the documents are usually very long and sparse even after pruning of the vocabulary. The process of lemmatization groups together different inflected forms of a word to a single instance (lemma) resulting in reduction of the dimensionality of the document vectors.

We used the Common Part-of-speech Tagger (COMPOST)[4] for lemmatization of Czech and WordNetLemmatizer from NLTK[5] to lemmatize English. Apart from the lemma we also used the information from the tagger to extract only nouns, adjectives, and verbs, to further reduce the dimensionality of the document vectors.

3.3. Database Postprocessing Statistics

The effect of the described preprocessing steps on the datasets are summarized in Table 1 in columns ADL, AUWD, and AVS. The statistics were calculated from 10 random samples from the datasets using stratified sampling with appropriate training set sizes, these are the statistical properties of the datasets seen by the different models during training. The training set sizes of the known datasets where chosen according to previous experiments made by others [4,7].

4. Feature Selection Methods

The classification performance of the models can be hugely influenced by the selection of words in the vocabulary. We searched for and decided to use two common feature (word) selection techniques: *information gain* (IG), *chi-square statistic* (χ^2) ([5,8,10]), that works well and one non-standard feature selection method known as *within class popularity* (WCP) reported to have a big improvement gap over the previous methods [9]. The impact of these techniques on the classification precision of the models is being part of our examination in the experiments.

[4]The Common POS Tagger - COMPOST has been developed by the Institute of Formal and Applied Linguistics, http://ufal.mff.cuni.cz/.

[5]NLTK Homepage: `http://www.nltk.org/`

Table 1. Database Postprocessing Statistics. Legend: Number of documents (ND), Minimum category size (mCS), Maximum category size (MCS), Average document length (ADL), average number of unique words per document (AUWD), average vocabulary size (AVS), number of categories (NC), and training set size (TSS). The 3 values reported in the columns ADL, AUWD, and AVS, are the average numbers of words in the vocabulary after: no processing, vocabulary pruning, and vocabulary pruning together with lemmatization.

Dataset	ND	mCS	MCS	ADL	AUWD	AVS	NC	TSS
20news	15076	503	800	279 112 111	142 76 74	93412 15652 14474	20	80%
sector	4795	14	53	352 183 182	155 99 97	48211 11388 10615	104	50%
webkb	2940	353	1149	273 129 130	144 82 82	33602 6864 6542	4	70%
seznam1k	916	2	56	733 590 567	375 345 297	87165 91115 59687	36	50%
seznam11k	8916	133	353	863 560 575	426 334 305	100000 41968 27702	36	80%

Note that the former two methods compute local word importances, i.e. how the word is "important" in the context of the document category, where on the contrary WCP computes the word importance globally, i.e. how the word is important in the context of the whole corpus. In order to assess the global importance for the two methods, we adopted the best performing approaches from [8], which are for IG and χ^2 the sum and a maximum over the local importances, respectively.

5. Experiments Description

The goal of the first experiments was to compare the performance of all the models trained on every document vector representation mentioned in Section 1. Our primary goal was to find out how the performance of the best classification models (found in training) are susceptible to different corpora preprocessing steps. We trained the models on corpora with pruned vocabulary (1), lemmatized pruned vocabulary (2), pruned vocabulary after feature selection (3), and lemmatized pruned vocabulary after feature selection (4), where the numbers in brackets are used as cross-reference into Figure 1 summarizing the results. When feature selection methods were applied, the words in the vocabulary were initially order by their importances and the final vocabulary was built out of only 10%, 20%, ... up to 90% of the most important words.

In our next experiments we were interested in finding what is the (relative) minimum number of words that can be selected from already pruned and lemmatized vocabulary using the feature selection methods without having substantial deteriorating effect on the performance of the classifiers. The results of these experiments are summarized in Figure 2.

In the both experiment setups we also compared the classifiers against (linear) Support Vector Machines in one-vs-all regime [2], which is to our knowledge considered the state-of-the-art text classifier.

We have implemented the classification models ourselves in Python except for SVM, in which case we used the open-source machine learning library for Python *scikit-learn*.

6. Classification Results

The performance of the models is measured using micro-averaged precision [1], defined as $\frac{\sum_{c=1}^{\mathcal{C}} TP_c}{\sum_{c=1}^{\mathcal{C}} TP_c + FP_c}$ where TP_c and FP_c is the number of documents correctly classified into category c and the number of documents incorrectly classified into category c.

We have chosen to optimize precision solely for the purpose of the target domain, which is categorization of web pages for advertising, where the cost of false positives, manifesting in displaying adverts not corresponding with the web page content or worse, being inappropriate for it, can be relatively high.

The results in Table 2, Figure 1, and Figure 2 are calculated from the averages of the results from the experiments (see Section 5) run repeatedly over 5 random splits of the datasets on training and hold-out sets (for training set sizes see Table 1). We took considerable care to construct vocabularies only from the training splits in effort not to leak any information from the hold-out set into training. Also MNB and CNB models were validated with different word smoothing factors [7] and SVM with different penalty factors.

Figure 1 illustrates not only how the best combination of the model together with a corpora transformation performs (the peaks of the bars) but also what is its performance gap from the *baseline* version of the classifier (shaded bar). Note that we omitted the results for DCM and CDCM in case of reduced vocabularies. These models are naturally suited for bigger vocabularies hence their results in this setup would be inadequate and meaningless.

Figure 1 and Table 2 which provide numerical values for the best performances of the models without feature selection (left side) and the best and worst (in brackets) performances of the models per feature selector (right side). It can be concluded that the reduction of vocabularies has not a substantial, yet not insignificant ($\leq 2\%$) impact on the performances on `webkb`, `seznam1k`, and `seznam11k` datasets, but on the contrary big vocabulary reduction in case of `20news` and `sector` leads to great loss in precision (up to 6% and 13%, respectively, on average across classifiers). We think that the reason for this is given by the textual content and diversity of the topics, it is certainly not by the external characteristics of the datasets, since the two are "complementary" in terms of the basic statistics (`sector` has fewer documents and a lot of categories with skewed distribution, on the contrary `20news` has more documents, fewer categories that are uniformly distributed, both have vocabularies of comparable sizes).

Figure 2 shows how severe the loss in performance of the models is for different degrees of reduction of the vocabulary. Following the discussion above, we reduced the vocabulary in case of `webkb`, `seznam1k`, and `seznam11k` to 1%, up to 9% to find when the performance start to drop similarly to `20news` and `sector`, and from how it looks we hit just the right range. We found that in datasets of our interest we can afford to lose up to 94% of the vocabulary without paying the price in terms of precision.

We conclude this section by stating that from the results on the different datasets none of the considered models matches the performance of SVM even with all the possible transforms and feature selections. On the other hand DCM and CDCM models performed worse, these models are hard to train and according to our result does not prove useful for a simple categorization task as ours. But considering the computational demands for training and demonstrated robustness, we conclude that the model of our choice is from the category of Multinomial Naive Bayes models.

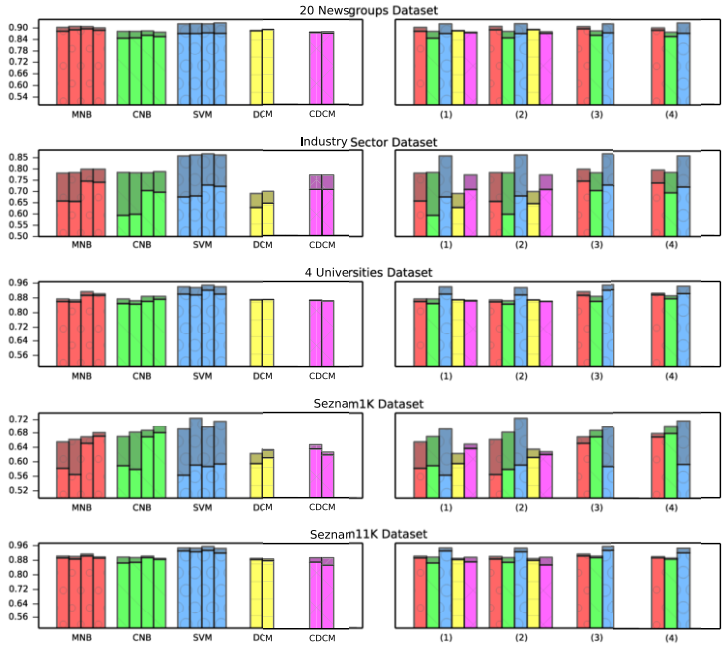


Figure 1. Impact of different preprocessing steps on the precision for individual classification models (*left*) and across different classification models (*right*).

Finally, since the results of the different classification models over different datasets and languages exhibit the same behaviour we conclude that we found no evidence that the performance of the models should be language dependent in case of English and Czech.

7. Conclusion

We presented results of an extensive empirical study of known Bayesian models and enhanced non-Bayesian alternatives to Multinomial Bayes models for text categorization. The study shows what is the best achievable performance for web documents classified to a given number of categories. The optimum combination of the model, the feature extraction method and preprocessing steps can be read out of the tables and graphs referred from Section 6.

We found that on two out of five datasets, among which is the Czech dataset of our interest, we can afford to reduce the vocabulary size approximately to 10% of the total without any classification performance degradation. This property holds across all types of models.

We also proved, that the models perform similarly across two different languages, English and Czech. Despite the fact that Czech is especially difficult and morphologically rich language, we found that lemmatization has no significant influence on classification precision of the considered models, but in the end it does not hurt and it can substantially reduce the vocabulary size which makes the computation during training and classification much faster.

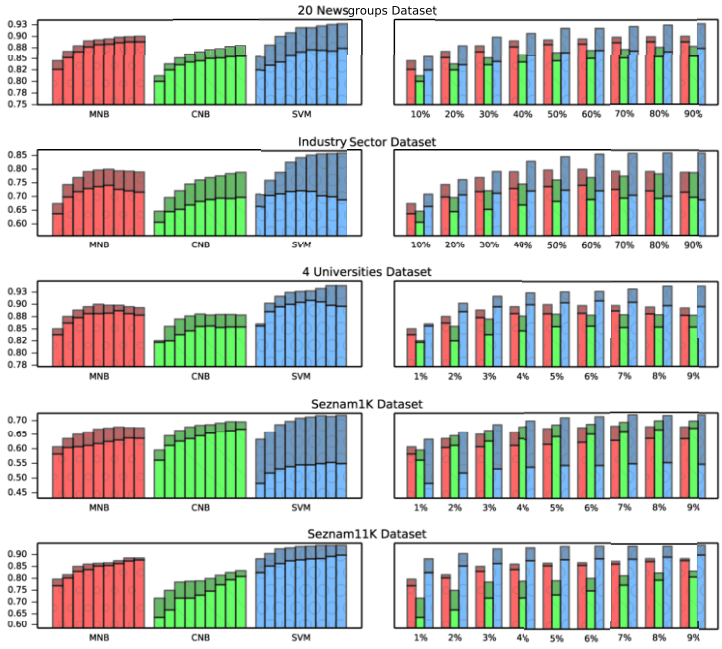


Figure 2. Impact of vocabulary reduction by feature selection methods on the precision for individual classification models (*left*) and across different classification models (*right*) for vocabulary reduced to 1%/10% up to 9%/90%.

Table 2. The maximum value of precision across different classification models and different feature selectors.

20 Newsgroups Dataset

MNB	CNB	SVM	DCM	CDCM	FS	MNB	CNB	SVM
0.91	0.88	0.92	0.89	0.88	IG	0.90 (0.83)	0.88 (0.80)	0.92 (0.84)
					CHI	0.90 (0.84)	0.89 (0.81)	0.93 (0.85)
					WCP	0.90 (0.85)	0.88 (0.81)	0.92 (0.86)

Industry Sector Dataset

MNB	CNB	SVM	DCM	CDCM	FS	MNB	CNB	SVM
0.78	0.78	0.86	0.70	0.77	IG	0.78 (0.60)	0.78 (0.52)	0.86 (0.71)
					CHI	0.80 (0.68)	0.78 (0.65)	0.86 (0.69)
					WCP	0.76 (0.60)	0.78 (0.51)	0.86 (0.70)

4 Universities Dataset

MNB	CNB	SVM	DCM	CDCM	FS	MNB	CNB	SVM
0.87	0.86	0.94	0.87	0.86	IG	0.89 (0.88)	0.88 (0.87)	0.94 (0.94)
					CHI	0.90 (0.88)	0.88 (0.87)	0.94 (0.94)
					WCP	0.90 (0.88)	0.87 (0.86)	0.94 (0.90)

Seznam1K Dataset

MNB	CNB	SVM	DCM	CDCM	FS	MNB	CNB	SVM
0.66	0.68	0.72	0.64	0.63	IG	0.68 (0.67)	0.70 (0.68)	0.72 (0.71)
					CHI	0.69 (0.65)	0.70 (0.65)	0.72 (0.65)
					WCP	0.68 (0.67)	0.69 (0.68)	0.72 (0.71)

Seznam11K Dataset

MNB	CNB	SVM	DCM	CDCM	FS	MNB	CNB	SVM
0.90	0.90	0.95	0.89	0.90	IG	0.89 (0.88)	0.90 (0.84)	0.95 (0.94)
					CHI	0.89 (0.89)	0.90 (0.82)	0.95 (0.93)
					WCP	0.89 (0.89)	0.90 (0.85)	0.95 (0.93)

Furthermore our results show that none of the feature selection techniques considered in this study is bringing significant improvement. Also, based on our empirical results, we can surely recommend the document vector representation $\boldsymbol{x}^{\mathbf{15}}$ (see Section 1.4), which is the only document vector representation that led most often (almost all the time) to improvement in the performance of the models.

To conclude, we came to similar result as previous studies that SVM is consistently delivering the top performance. It is certainly the best choice when the training computational requirements are not of a concern, training SVM model in comparison with training of MNB model, which is as easy as simple counting, is a completely different (quadratic optimization) story. Yet the runtime number of operations for all considered models is linearly dependent on the size of the vocabulary and the number of categories.

Finally, note that the best MNB is performing almost on par with SVM, which means in practice, when the best classification performance is not necessary, picking up the right configuration for MNB is probably the right way to go.

Acknowledgements

This work was supported by the Decision Making and Control for Manufacturing III, Research programme MSM 6840770038, funded by the Czech Ministry of Education.

References

[1] George Forman. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3:1289–1305, March 2003.

[2] Thorsten Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 137–142, London, UK, UK, 1998. Springer-Verlag.

[3] Ashraf M. Kibriya, Eibe Frank, Bernhard Pfahringer, and Geoffrey Holmes. Multinomial naive bayes for text categorization revisited. In *Proceedings of the 17th Australian Joint Conference on Advances in Artificial Intelligence*, AI'04, pages 488–499, Berlin, Heidelberg, 2004. Springer-Verlag.

[4] Rasmus E. Madsen, David Kauchak, and Charles Elkan. Modeling word burstiness using the dirichlet distribution. In *In Proceedings of the 22nd international conference on Machine learning, volume 119 of ACM International Conference Proceeding Series*, pages 545–552, 2005.

[5] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[6] Thomas P. Minka. Estimating a dirichlet distribution. Technical report, 2000.

[7] Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *In Proceedings of the Twentieth International Conference on Machine Learning*, pages 616–623, 2003.

[8] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, March 2002.

[9] Sanasam Ranbir Singh, Hema A. Murthy, and Timothy A. Gonsalves. Feature selection for text classification based on gini coefficient of inequality. In Huan Liu, Hiroshi Motoda, Rudy Setiono, and Zheng Zhao, editors, *FSDM*, volume 10 of *JMLR Proceedings*, pages 76–85. JMLR.org, 2010.

[10] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

This page intentionally left blank

STAIRS 2014
U. Endriss and J. Leite (Eds.)

Subject Index

U. Endriss and J. Leite (Eds.)

Author Index

This page intentionally left blank