Research and Development in Intelligent Systems XXVII

Max Bramer · Miltos Petridis · Adrian Hopgood
Editors

Research and Development in Intelligent Systems XXVII

Incorporating Applications and Innovations in Intelligent Systems XVIII

Proceedings of AI-2010, The Thirtieth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence

Springer

Editors
Prof. Max Bramer
University of Portsmouth
UK

Miltos Petridis
University of Greenwich
UK

Adrian Hopgood
De Montford University
Leicester

ISBN 978-0-85729-129-5 e-ISBN 978-0-85729-130-1
DOI 10.1007/978-0-85729-130-1
Springer London Dordrecht Heidelberg New York

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Printed on acid-free paper

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

PROGRAMME CHAIRS' INTRODUCTION

M.A.BRAMER, University of Portsmouth, UK
M.PETRIDIS, University of Greenwich, UK

This volume comprises the refereed papers presented at AI-2010, the Thirtieth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, held in Cambridge in December 2010 in both the technical and the application streams. The conference was organised by SGAI, the British Computer Society Specialist Group on Artificial Intelligence.

The technical papers included new and innovative developments in the field, divided into sections on Intelligent Agents, Knowledge Discovery and Data Mining, Evolutionary Algorithms, Bayesian Networks and Model-Based Diagnosis, Machine Learning and Planning and Scheduling.

This year's Donald Michie Memorial Award for the best refereed technical paper was won by a paper entitled "Effective Product Recommendation using the Real-Time Web" by S. Garcia Esparza, M. P. O'Mahony and B. Smyth (University College Dublin, Ireland).

The application papers included present innovative applications of AI techniques in a number of subject domains. This year, the papers are divided into sections on Applications of Machine Learning I and II, AI for Scheduling and AI in Action.

This year's Rob Milne Memorial Award for the best refereed application paper was won by a paper entitled "Artificial Intelligence Techniques for the Berth Allocation and Container Stacking Problems in Container Terminals" by Miguel A. Salido, Mario Rodriguez-Molins and Federico Barber (Technical University of Valencia, Spain).

The volume also includes the text of short papers presented as posters at the conference.

On behalf of the conference organising committee we would like to thank all those who contributed to the organisation of this year's programme, in particular the programme committee members, the executive programme committees and our administrators Rachel Browning and Bryony Bramer.

Max Bramer, Technical Programme Chair, AI-2010
Miltos Petridis, Application Programme Chair, AI-2010

ACKNOWLEDGEMENTS

AI-2010 CONFERENCE COMMITTEE

Prof. Adrian Hopgood De Montfort University	(Conference Chair)
Prof. Max Bramer University of Portsmouth	(Technical Programme Chair)
Dr. Miltos Petridis University of Greenwich	(Application Programme Chair and UK CBR Organiser)
Dr. David Elizondo De Montfort University	(Workshop Organiser)
Rosemary Gilligan	(Treasurer)
Dr Nirmalie Wiratunga The Robert Gordon University	(Poster Session Organiser)
Dr. Alice Kerly University of Birmingham	(Research Student Liaison)
Dr. Kirsty Bradbrook	(Research Student Liaison)
Rachel Browning BCS	(Conference Administrator)
Bryony Bramer	(Paper Administrator)

TECHNICAL EXECUTIVE PROGRAMME COMMITTEE

Prof. Max Bramer, University of Portsmouth (Chair)

Dr. John Kingston, Health & Safety Laboratory

Dr. Peter Lucas, University of Nijmegen, The Netherlands

Dr. Nirmalie Wiratunga, The Robert Gordon University, Aberdeen

APPLICATIONS EXECUTIVE PROGRAMME COMMITTEE

Dr. Miltos Petridis, University of Greenwich (Chair)

Mr. Richard Ellis, Helyx

Ms. Rosemary Gilligan

Dr. Richard Wheeler, University of Edinburgh

TECHNICAL PROGRAMME COMMITTEE

APPLICATION PROGRAMME COMMITTEE

Hatem Ahriz (Robert Gordon University)

Tony Allen (Nottingham Trent University)

Ines Arana (Robert Gordon University)

Mercedes Argüello Casteleiro (University of Manchester)

Kirsty Bradbrook (Vtesse Networks Ltd)

Ken Brown (University College Cork)

Simon Coupland (De Montfort University)

Sarah Jane Delany (Dublin Institute of Technology)

Richard Ellis (Helyx)

Lindsay Evett (Nottingham Trent University)

Rosemary Gilligan (University of Hertfordshire)

John Gordon (AKRI Ltd)

Elizabeth Guest (Leeds Metropolitan University)

Chris Hinde (Loughborough University)

Adrian Hopgood (De Montfort University)

Alice Kerly (Selex Systems Integration Ltd)

Shuliang Li (University of Westminster)

Jixin Ma (University of Greenwich)

Lars Nolle (Nottingham Trent University)

Miltos Petridis (University of Greenwich)

Rong Qu (University of Nottingham)

Miguel Salido (Universidad Politécnica de Valencia)

Roger Tait (University of Cambridge)

Wamberto Vasconcelos (University of Aberdeen)

Richard Wheeler (Human Computer Learning Foundation)

Patrick Wong (Open University)

CONTENTS

Research and Development in Intelligent Systems XXVII

PLANNING AND SCHEDULING

MACHINE LEARNING

EVOLUTIONARY ALGORITHMS, BAYESIAN NETWORKS AND MODEL-BASED DIAGNOSIS

SHORT PAPERS

Applications and Innovations in Intelligent Systems XVIII

BEST APPLICATION PAPER

APPLICATIONS OF MACHINE LEARNING I

APPLICATIONS OF MACHINE LEARNING II

AI FOR SCHEDULING

AI IN ACTION

SHORT PAPERS

Research and Development in Intelligent Systems XXVII

BEST TECHNICAL PAPER

Effective Product Recommendation Using the Real-Time Web

Sandra Garcia Esparza, Michael P. O'Mahony and Barry Smyth

Abstract The so-called real-time web (RTW) is a web of opinions, comments, and personal viewpoints, often expressed in the form of short, 140-character text messages providing abbreviated and highly personalized commentary in real-time. Today, Twitter is undoubtedly the king of the RTW. It boasts 190 million users and generates in the region of 65m tweets per day[1]. This RTW data is far from the structured data (movie ratings, product features, etc.) that is familiar to recommender systems research but it is useful to consider its applicability to recommendation scenarios. In this paper we consider harnessing the real-time opinions of users, expressed through the Twitter-like short textual reviews available on the Blippr service (www.blippr.com). In particular we describe how users and products can be represented from the terms used in their associated reviews and describe experiments to highlight the recommendation potential of this RTW data-source and approach.

1 Introduction

Recommender systems have proven to be an important way for people to discover information, products and services that are relevant to their needs. Recommender systems complement the more conventional query-based search services by offering more proactive information discovery, often based on a profile of users' short- or long-term preferences. It is useful to view many recommendation techniques as falling, broadly speaking, into one of two basic categories: *collaborative filtering* versus *content-based* approaches.

Sandra Garcia Esparza, Michael P. O'Mahony and Barry Smyth
CLARITY: Centre for Sensor Web Technologies, School of Computer Science and Informatics, University College Dublin, Ireland.
e-mail: {sandra.garcia-esparza,michael.omahony,barry.smyth}@ucd.ie

[1] http://techcrunch.com/2010/06/08/twitter-190-million-users/

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_1,

In collaborative filtering approaches, items are selected for recommendation to some target user based on the items that similar users have liked in the past [24]. The key source of recommendation knowledge that collaborative filtering approaches use is the *ratings matrix*. This is a *user-item* matrix that captures the *interest* that a user U_i has in item I_j. Sometimes these interests are in the form of explicit ratings; for example, in MovieLens[2] users express their movie interests on the basis of a 1-5 rating scale. Other times these interests can be inferred from user actions; for example, Amazon's recommendations are based on user transaction histories and in this sense the purchasing of an item is viewed as a strongly positive rating. Very briefly, there are two flavours of collaborative filtering: (1) *user-based* techniques [18, 24] generate recommendations for a target user based on the items that similar users (that is, similarity among the rows of the ratings matrix) have liked in the past; (2) *item-based* approaches [21] generate recommendations based on the items that are similar to the items (that is, similarity among the columns of the ratings matrix) that the target user has liked in the past. Recent years has seen considerable research effort invested into this form of recommendation technique; in particular, focusing the manipulation of the core ratings matrix to better identify latent interests as a source of recommendation knowledge [9, 10].

Collaborative filtering approaches have been shown to work well when there is sufficient information to populate the ratings matrix, but very often this matrix is sparsely populated leading to poor coverage of the recommendation space and ultimately limiting recommendation effectiveness [2]. The alternative *content-based* approach to recommendation avoids the need for user ratings data, drawing instead on more richly detailed content representations of the items to be recommended [4]. For example, meta-data about a movie (genre, director, actors, etc.) can be used as the basis for item-level similarity assessment allowing content-based recommenders to rank items that are similar (content-wise) to the items that a target user is known to like (and perhaps dissimilar to the items that the target user is known to dislike). Content-based approaches have been used in a variety of recommendation applications including TV, e-commerce and travel [6, 22, 25]. In addition, researchers have looked at the potential to combine collaborative filtering and content based approaches as the basis for *hybrid* recommendation strategies [5]. A key challenge, however, relating to content-based systems is the overhead involved in obtaining the meta-data required to represent items; indeed, for some domains (e.g. jokes, works of art etc.), representing items effectively with such data can be problematic.

There is, however, a third source of recommendation data that can be considered. Most readers will be familiar with Twitter's short-form text messages (*tweets*), that allow users to broadcast their opinions on life, the universe and everything to just about anyone who cares to listen. Sometimes these messages carry important preference-like information or even a product review; for example, one recent new iPad owner posted: *"Typing this tweet on iPad. I love it. With wireless keyboard I could see this as my laptop replacement."* This tweet is clearly expressing a positive opinion on Apple's latest creation. Moreover, this type of 'review' carries some im-

[2] `http://www.grouplens.org`

portant recommendation information and not just simple sentiment, but also specific information about certain features (in this case, the wireless keyboard). Already researchers and practitioners alike have begun to enthuse about the potential for this type of user-generated content to influence the marketing of products and services [8]. Our interests run deeper, and in this paper we explore whether these fragmented and noisy snippets of user opinions can be used more directly in recommendation. To this end we consider two important questions: (1) Can RTW data be used as the basis for representing, indexing, and recommending items, products and services? (2) How well does a recommender system based on RTW data perform relative to traditional approaches? In what follows we describe experiments that are designed to shed light on these important questions. Specifically, we develop a product recommender system that is powered by Twitter-like product-related comments and show that it has the potential to outperform a comparable collaborative filtering approach.

The paper is organized as follows. In Section 2, we describe related work that has been carried out on sentiment analysis and opinion mining of user-generated content. A description of the Blippr service[3], which we use as our test domain, is presented in Section 3. Our recommender approach, based on RTW data, is detailed in Section 4 and the results of an empirical evaluation of the approach are given in Section 5. Finally, we present concluding remarks in Section 6.

2 Related Work

In past years, user opinions in the form of reviews, comments, blogs and microblogs have been used by researchers for different purposes. One of the areas which has captured the interest of researchers is the application of sentiment analysis techniques to these opinions. In addition, user-generated content has also served as an additional source of knowledge for recommender systems. Here, we provide an overview of some of the work that has been carried out in this regard.

Sentiment analysis [26] encompasses areas such as subjectivity classification [28], opinion summarization [7] and review rating prediction [30]. Traditional text classification techniques based on machine learning have been applied in sentiment classification and indeed have proven their efficiency in many occasions [12, 14]. However, in [17] it is demonstrated how these models are topic-dependant, domain-dependant and temporally-dependant. Moreover, they suggest that relying on the emoticons present in the text — and using those texts as training data — can be a way of reducing the previous dependencies.

Lately, sentiment analysis techniques have also been applied to short texts like micro-blog messages. In [13], the authors present different machine learning techniques to classify Twitter messages as positive, negative or neutral. In order to do so, they create two classifiers: a neutral-sentiment classifier and a polarity (negative or positive) classifier. Extracting product features from reviews and identify-

[3] http://www.blippr.com

ing opinions associated with these features has also been studied in [16]. Our approach, however, is not aimed at employing sentiment analysis or opinion mining techniques; instead, we are interested in using user-generated content to provide better recommendations than traditional recommender systems.

Indeed, researchers have recently begun to consider the utility of such content as an additional source of recommendation data. For example, the role of tags in recommender systems has been examined in [23]. Further, researchers have started to leverage user-generated reviews as a way to recommend and filter products and services. For example, in [11, 15] the number of ratings in a collaborative filtering system is increased by inferring new ratings from user reviews using sentiment analysis techniques. Both works are evaluated on movie datasets (the former on Netflix and Flixster and the latter on IMDb).

In [29], another example is presented where a recommender system avails of user-generated content. They propose a hybrid collaborative filtering and content-based approach to recommend hotels and attractions, where the collaborative filtering component benefits from user-generated reviews. Moreover, they also comment on the advantages of using user-generated content for recommender systems; such as, for example, providing a better rationale for recommended products and increasing user trust in the system. Similar ideas are presented in [1], which look at using user-generated movie reviews from IMDb in combination with movie metadata (e.g. keywords, genres, plot outlines and synopses) as input for a movie recommender system. Their results show that user reviews provide the best source of information for movie recommendations, followed by movie genre data.

The approach proposed in this paper expands on the above work. In particular, our approach to product recommendation involves representing users and products based on the terms used in associated reviews, from which recommendations are subsequently made. In addition, we focus on short reviews from micro-blogging services as opposed to the longer-form product reviews that have typically been considered in previous work. In the next section, we describe the Blippr service, from where the micro-review data that is employed in our approach is sourced.

3 The Blippr Service

In this paper we focus on a Twitter-like review service called Blippr. This service allows registered users to review products from five different categories: *applications*, *music*, *movies*, *books* and *games*. These reviews (or *blips*) are in the form of 160-character text messages, and users must also supply an accompanying rating on a 4-point rating scale: *love it*, *like it*, *dislike it* or *hate it*. For instance, Figure 1 shows a screenshot of the Blippr interface when a user wants to add a new blip about the movie *The Matrix*. The user must add a review and a rating. In addition, the website shows past reviews for this movie from other users and their associated ratings.

Besides adding blips, users can also add tags to products. However, in order to avoid user abuse, Blippr currently does not allow users to tag popular products nor

Fig. 1 Adding a blip about a movie on the Blippr service.

to see which users added particular tags. Blippr also provides users with recommendations for the different product types, although precise details on the recommendation algorithm employed have not been published. Further, Blippr users can follow friends in a Twitter-like fashion and share their reviews with them. Finally, users can also post their blips to other services like Twitter or Buzz.

The Blippr service provides us with a useful source of real-time web data, which facilitates an analysis of the performance of recommendation algorithms across a range of product types. In the next section, we describe our recommendation techniques in detail and show how the micro-blogging activity of users can be harnessed to deliver effective product recommendations.

4 Product Recommendation using RTW Data

A key issue with collaborative and content-based recommenders is that oftentimes neither user ratings nor item meta-data are available to effectively drive either approach. In this paper, we explore a third source of recommendation data — namely, user-generated content relating to products and services — to deal with such situations. While user-generated content is inherently noisy, it is plentiful and here we describe an approach which uses this data in order to recommend products to users.

4.1 Index Creation

Our approach involves the creation of two indices, representing users and products, from which product recommendations are made to users. In this section, we consider how real-time web data can be used as a source of indexing information.

Product Index. We create this index as follows. Consider a product P_i which is associated with a set of blips and tags as per Equation 1. In turn, each blip is made up of a set of terms and so each product can be represented as a set of terms (drawn from blips and tags) using a bag-of-words style approach [19] according to Equation 1.

$$P_i = \{b_1, ..., b_k\} \cup \{tag_1, ...tag_m\} = \{t_1, ..., t_n\} \tag{1}$$

In this way individual products can be viewed as documents made up of the set of terms (words) contained in their associated blips and tags. We can create an index of these documents so that we can retrieve documents (that is products) based on the terms that are present in their blips and tags. The information retrieval community provides a well understood set of techniques for dealing with just this form of document representation and retrieval. For example, there are many ways to *weight* the terms that are associated with a given product based on how representative or informative these terms are with respect to the product in question. Here we use the well known TFIDF approach [19] to term weighting (Equation 2). Briefly, the weight of a term t_j in a product P_i, with respect to some collection of products $\mathbf{P}$, is proportional to the frequency of occurrence of t_j in P_i (denoted by n_{t_j,P_i}), but inversely proportional to the frequency of occurrence of t_j in $\mathbf{P}$ overall, thus giving preference to terms that help to discriminate P_i from the other products in the collection.

$$\text{TFIDF}(P_i, t_j, \mathbf{P}) = \frac{n_{t_j,P_i}}{\sum_{t_k \in P_i} n_{t_k,P_i}} \times \log\left(\frac{|\mathbf{P}|}{|\{P_k \in \mathbf{P} : t_j \in P_k\}|}\right) \tag{2}$$

Thus we can create a term-based index of products $\mathbf{P}$, such that each entry $\mathbf{P_{ij}}$ encodes the importance of term t_j in product P_i; see Equation 3. In this work we use Lucene[4] to provide this indexing and term-weighting functionality.

$$\mathbf{P_{ij}} = \text{TFIDF}(P_i, t_j, \mathbf{P}) \tag{3}$$

User Index. We use a similar approach to that above to create the user index. Specifically, we treat each user as a document made up of their blips (Equation 4); since we could not obtain the tags submitted by individual users from Blippr, it is not possible to represent users by tags. As before, we index the set of users using Lucene to produce a user index, $\mathbf{U}$, such that each entry $\mathbf{U_{ij}}$ encodes the importance of term t_j for user U_i, once again using Lucene's TFIDF scoring function as per Equation 5.

$$U_i = \{b_1, ..., b_k\} = \{t_1, ..., t_n\} \tag{4}$$

$$\mathbf{U_{ij}} = \text{TFIDF}(U_i, t_j, \mathbf{U}) \tag{5}$$

[4] `http://lucene.apache.org`

```
Input: Target user U_T, user index U, product index P, number of
products to retrieve n
Output: Top n product recommendations

1.  UserBasedRecommendation (U_T, U, P, n)
2.  Begin
3.      query ← U.get(U_T)          // Return term vector for U_T in U
4.      recs ← P.retrieve(query)    // Retrieve ranked list of
                                    // products from P based on query
5.      return recs.first(n)        // Return top n recommendations
6.  End
```

Fig. 2 User-based recommendation algorithm.

4.2 Recommending Products

In the above, we have described how two types of index for use in recommendation are created: an index of users, based on the terms in their blips, and an index of products, based on the terms in their blips (or in their tags or the combination of blips and tags). This suggests the following recommendation strategies. First, we can implement a *user-based* approach in which the *target user's* profile acts as a query against the product index to produce a ranked-list of similar products[5]; see Figure 2. We consider three variations on this approach, the first based on a product index of blips (B), the second based on a product index of tags (T), and the third based on a product index of blips and tags ($B+T$).

In addition, to provide a benchmark for the above approach, we implement a *community-based* approach based on collaborative filtering ideas [24]. We identify a set of similar users, by using the target user profile as a query on the user index, and then rank the preferred products of these similar users based on their frequency of occurrence in the similar user profiles; see Figure 3. We can adjust this algorithm by retrieving different numbers of similar users; in Section 5 we compare the retrieval performance provided by using 10 and 100 nearest neighbours.

5 Evaluation

We now evaluate the recommendation performance provided by the RTW-based algorithms described above. We begin by describing the datasets used in our evaluation and the metrics that we employ to measure performance.

[5] The target user's blips are first removed from the product index to ensure that no bias is introduced into the process.

Input: Target user U_T, user index **U**, product index **P**, number of products to retrieve n, neighbourhood size k
Output: Top n movie recommendations

```
1.  CommunityBasedRecommendation (U_T, U, P, n, k)
2.  Begin
3.      query ← U.get(U_T)            // Return term vector for U_T in U
4.      users ← U.retrieve(query)     // Get ranked list of similar users
5.      neighs ← users.first(k)       // Get the top k most similar
                                      // users as neighbours
6.      recs ← {}                     // Get all neighbours' products
7.      for each n ∈ neighs
8.          recs ← recs ∪ n.products()
9.      end

10.     return recs.sort(score(.,.), n)   // Return top n most frequently
                                          // occurring products
                                          // score(P_i, neighs) = ∑_{n∈neighs} occurs(P_i, n)
11. end
```

Fig. 3 Community-based recommendation algorithm.

5.1 Datasets

Our experiments use Blippr data relating to 4 different product types: *movies*, *books*, *applications* (apps) and *games*. As previously mentioned, Blippr facilitates feedback on items from 5 product types; in our work, we do not consider *music* products due to the small number of blips of this product type. For clarity, we focus on strong-positive blips only (i.e. where users have expressed the highest sentiment toward items). We collected data from the website using the Blippr API in April 2010, capturing blips written before that date (other data had to be scraped from the website due to the limitations of the API). We performed some preprocessing on the extracted blips such as removing stopwords, special symbols (?, &,*, etc.), digits and multiple repetitions of characters in words (e.g. we reduce *goooood* to *good*). Then we consider only blips that are written in English. For our experiments, we have selected those items that have received at least 3 blips and those users that have authored between 5 and 20 blips, inclusive. Dataset statistics are shown in Table 1.

Table 1 Statistics showing the number of items, tags and users present in each dataset.

Measure	Movies	Apps	Books	Games
# Items	1,080	268	313	277
# Users	542	373	120	164
# Blips	15,121	10,910	3,003	3,472
# Distinct Tags	1,543	817	649	165
Total # Tags	8,444	1,672	2,236	368

5.2 Metrics

We use *precision* and *recall*, which have been widely used in the field of information retrieval, to evaluate recommendation accuracy. These metrics have been adapted to evaluate the accuracy of a set of recommended items [20] and are defined as follows:

$$\text{Precision} = \frac{|T \cap R|}{|R|}, \qquad \text{Recall} = \frac{|T \cap R|}{|T|}, \tag{6}$$

where T is the test set and R is the recommended set of items for each user, respectively. Here, the test set for each user is given by the set of items that the user has blipped about (i.e. strong-positive blips of the user).

Precision and recall are often conflicting properties. For example, increasing the recommendation set size is likely to improve recall, but reduce precision. To resolve this conflict, we use the *F1 metric*, which is the harmonic mean of precision and recall [20, 27]. It is given by:

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{7}$$

We also evaluate recommendation *coverage*, which measures the number of products for which a recommender is capable of making recommendations (as a percentage of the total number of products in the system). Clearly, the ability to make recommendations for as many products as possible is a desirable system property.

5.3 Recommendation Results

To evaluate our recommendation algorithms, we first create separate product and user indices for each of the 4 datasets according to the approach described in Section 4. For each dataset, we consider each user in turn from the user index to act as a target user, U_T, as per Section 4.2 and compute precision, recall and F1 metric scores for different recommendation-list sizes ranging from 5 to 30 movies.

Precision and recall results are presented in Figures 4–7 (left) for the *movies*, *applications*, *books* and *games* datasets, respectively. For all datasets, there is a clear benefit for two of the user-based recommendation strategies (B and $B+T$) compared to the community-based approaches. Indexing products using blips and tags, however, does not provide improved recommendation performance over an index based on blips alone; adding tags to the blip-based index achieves little or no effect. For example, in the case of the *books* dataset using recommendation lists of size 5, we see that both user-based approaches enjoy a precision score of approximately 0.34, indicating that, on average, almost 2 of the 5 recommended books are known to be liked by target users.

In all cases, an index based on tags alone (T) provides the worst recommendation performance. We also carried out experiments adding meta-data (e.g. title of the

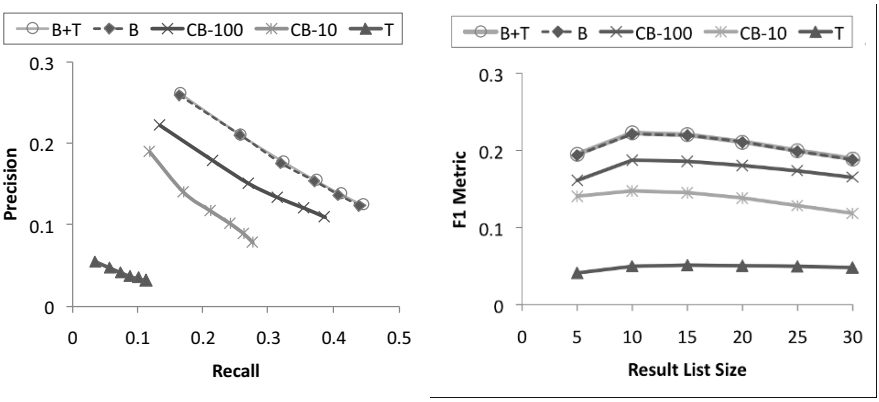


Fig. 4 *Movies* dataset: precision-recall (left) and F1 metric (right) for user-based (B vs. T vs. $B+T$) and community-based recommendation (CB-10 vs. CB-100).

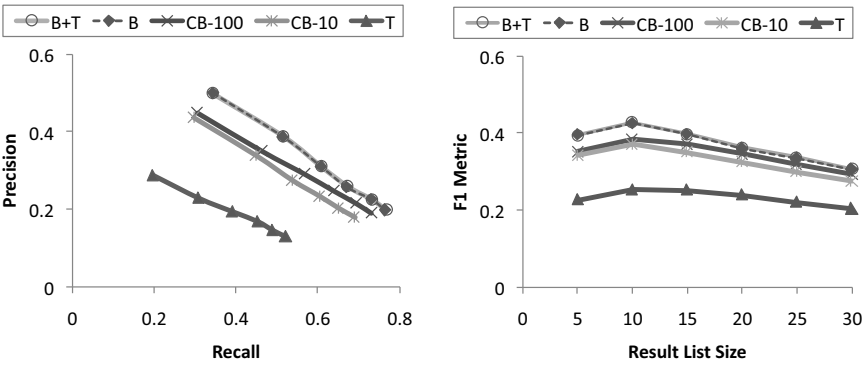


Fig. 5 *Applications* dataset: precision-recall (left) and F1 metric (right) for user-based (B vs. T vs. $B+T$) and community-based recommendation (CB-10 vs. CB-100).

product, genres, movie actors and directors, book authors and game platforms and developers) to the tag indices. Although some improvement in performance is seen using this approach for the tag index, overall the performance is still significantly worse compared to the other strategies and, in addition, the performance of indices based on blips and tags is not improved. We note, however, that tags and metadata may provide greater potential for recommendation in other domains, given the restrictions placed on adding tags to popular products on Blippr (see Section 3) and the relatively small numbers of tags present in our evaluation datasets.

Figures 4–7 (left) also show the community-based results when 10 (CB-10) and 100 (CB-100) similar users are selected as the basis for recommendation. For all except the *books* dataset, there is clearly a benefit when it comes to drawing on a larger community of similar users, although our tests suggest that this does not extend beyond 100 users in practice, and neither approach is able to match the precision and

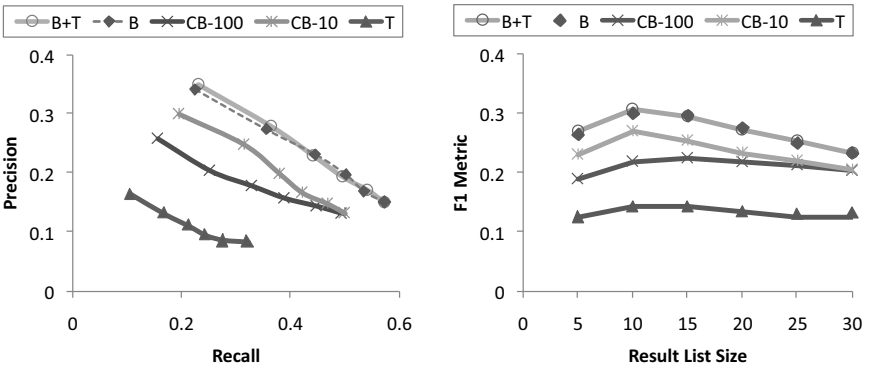


Fig. 6 *Books* dataset: precision-recall (left) and F1 metric (right) for user-based (B vs. T vs. $B+T$) and community-based recommendation (CB-10 vs. CB-100).

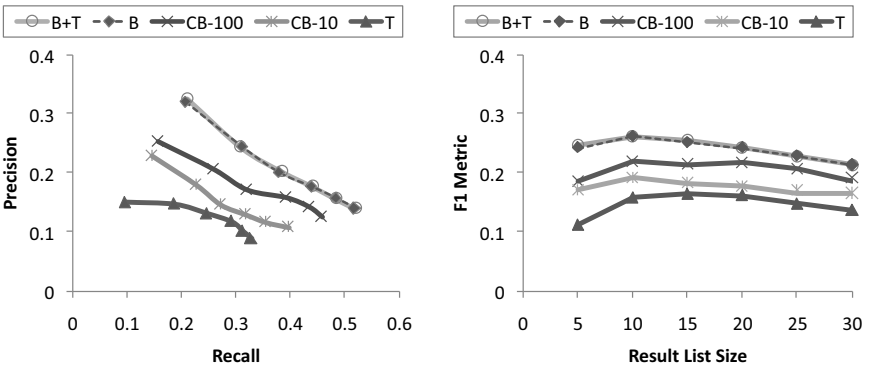


Fig. 7 *Games* dataset: precision-recall (left) and F1 metric (right) for user-based (B vs. T vs. $B+T$) and community-based recommendation (CB-10 vs. CB-100).

recall scores of the user-based strategies. The *books* dataset is the exception to this trend, where selecting 10 similar users achieves better performance than selecting 100 users (but did not outperform the blip-based index approach). This is likely due to the small number of users in this dataset; for example, the *books* dataset contains 120 users, compared to 542 users in the largest dataset (*movies*).

The F1 scores achieved by the 5 recommendation strategies are shown in Figures 4–7 (right). Obviously we see the same relative ordering of the different strategies as before with, for example, the user-based approach using a blip-based index delivering the best performance for all datasets. Interestingly, we also see that F1 is maximized for result-lists of size 10, indicating that the best balance of precision and recall is achieved for typical recommendation list sizes.

In Figure 8 (left), we compare the precision and recall performance provided by user-based recommendation using a blip-based index across the 4 datasets. It can

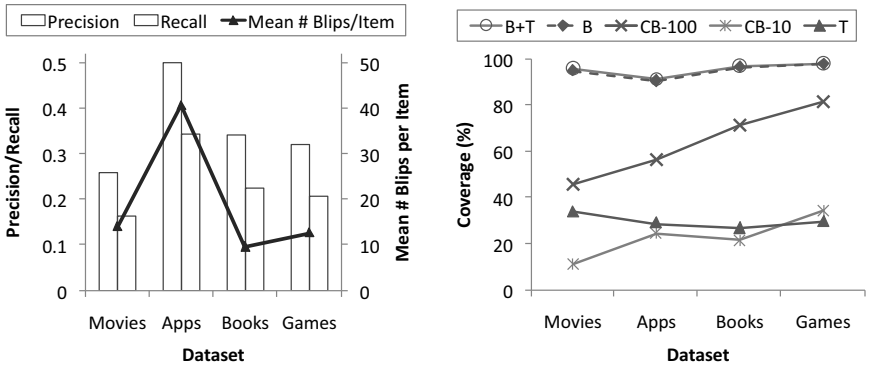


Fig. 8 Precision and recall (recommendation-list size = 5) provided by user-based recommendation using blip-based indices and mean number of blips per item for each dataset (left) and the coverage provided by the recommendation strategies for each dataset (right).

be seen that best performance is achieved for the *applications* dataset, with approximately similar trends observed for the other datasets. For example, precision and recall values of 0.50 and 0.34 are achieved for the *applications* dataset, respectively, compared to values of 0.34 and 0.23 for the *books* dataset (these values correspond to a recommendation list size of 5). Also shown in this figure is the mean number of blips per item for each dataset; it can be seen that these values correlate well with the precision (Pearson $r = 0.89$) and recall (Pearson $r = 0.90$) performance achieved for the datasets. This seems a reasonable finding, since it indicates that richer product indices (i.e. products which are described by a greater number of blips) lead to better recommendation performance. However, we note that the datasets used in our evaluation contain relatively small numbers of users, items and blips, and hence further analysis is required to make definitive conclusions in this regard.

Finally, we examine coverage performance in Figure 8 (right). The trends show that two of the user-based recommendation strategies (B and $B+T$) provide almost complete coverage for all datasets, well in excess of that given by both community-based approaches (even when using 100 nearest neighbours) and indexing by tags alone. These are very positive findings in respect of the utility of blips as a source of recommendation data, since they indicate that this approach is capable of providing significantly better coverage compared to the traditional community-based strategies, while being able to deliver more accurate recommendations as well.

6 Conclusions

In this paper we are interested in whether user-generated, micro-blogging messages, short and noisy as they are, have a role to play in recommender systems. We have described how to represent users and items based on micro-blogging reviews of 4

product types and tested this technique using a number of recommendation strategies on live-user data. The results are promising. They suggest that micro-blogging messages can provide a useful recommendation signal, despite their short-form and inconsistent use of language; we have found that indices based on blips outperform a more traditional collaborative-filtering based approach in all the datasets evaluated.

This work is novel in its use of micro-blogging information for recommendation. Our approach is related to a growing body of research on the potential for user-generated content to inform recommendation [1, 3, 29]. This related research focuses mainly on more conventional, long-form user reviews, whereas the work presented in this paper focuses on the more challenging micro-blogging messages. In future work, we will apply our approach to other domains like Twitter, which offers a rich source of user opinions on heterogeneous topics and products. In addition, we will expand our approach to recommend additional objects to users such as tags and other like-minded users and also consider the potential for cross-domain recommendation, where indices created using messages from one domain can be used to recommend products in other domains.

Acknowledgements Based on work supported by Science Foundation Ireland, Grant No. 07/CE/I1147.

References

1. S. Aciar, D. Zhang, S. Simoff, and J. Debenham. Recommender system based on consumer product reviews. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 719–723, Washington, DC, USA, 2006. IEEE Computer Society.
2. G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.
3. S. Ahn and C.-K. Shi. Exploring movie recommendation system using cultural metadata. *Transactions on Edutainment II*, pages 119–134, 2009.
4. M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.
5. R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
6. S. Chelcea, G. Gallais, and B. Trousse. A personalized recommender system for travel information. In *Proceedings of the 1st French-speaking conference on Mobility and ubiquity computing (UbiMob '04)*, pages 143–150, New York, NY, USA, 2004. ACM.
7. M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '04)*, pages 168–177, New York, NY, USA, 2004. ACM.
8. B. J. Jansen, M. Zhang, K. Sobel, and A. Chowdury. Micro-blogging as online word of mouth branding. *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems (CHI EA '09)*, pages 3859–3864, 2009.
9. Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 447–456, Paris, France, June 28–July 1 2009.
10. Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

11. C. W.-k. Leung, S. C.-f. Chan, and F.-l. Chung. Integrating collaborative filtering and sentiment analysis: A rating inference approach. In *Proceedings of the ECAI 2006 Workshop on Recommender Systems*, pages 62–66, Riva del Garda, Italy, 2006.
12. T. Mullen and N. Collier. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of the conference on Empirical Methods in Natural Language Processing*, 2004.
13. V. Pandey and C. Iyer. Sentiment analysis of microblogs. `http://www.stanford.edu/class/cs229/proj2009/PandeyIyer.pdf`, 2009. Accessed on: April 2010.
14. B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing (EMNLP '02)*, pages 79–86, Morristown, NJ, USA, 2002. Association for Computational Linguistics.
15. D. Poirier, I. Tellier, F. Franoise, and S. Julien. Toward text-based recommendations. In *Proceedings of the 9th international conference on Adaptivity, Personalization and Fusion of Heterogeneous Information (RIAO '10)*, Paris, France, 2010.
16. A.-M. Popescu and O. Etzioni. Extracting product features and opinions from reviews. *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT '05)*, pages 339–346, 2005.
17. J. Read. Using emoticons to reduce dependency in machine learning techniques for sentiment classification, 2005.
18. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM Conference on Computer-Supported Cooperative Work (CSCW 94)*, pages 175–186, Chapel Hill, North Carolina, USA, August 1994.
19. G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.
20. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for ecommerce. In *Proceedings of the 2nd ACM Conference on Electronic Commerce (EC '00)*, pages 158–167, Minneapolis, Minnesota, USA, October 17-20 2000. ACM.
21. B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International World Wide Web Conference (WWW '01)*, pages 285–295, Hong Kong, May 2001.
22. J. B. Schafer, J. A. Konstan, and J. Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1-2):115–153, 2001.
23. S. Sen, J. Vig, and J. Riedl. Tagommenders: connecting users to items through tags. In *Proceedings of the 18th international conference on World wide web (WWW '09)*, pages 671–680, New York, NY, USA, 2009. ACM.
24. U. Shardanand and P. Maes. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '95)*, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.
25. B. Smyth and P. Cotter. A personalised TV listings service for the digital TV age. *Knowledge-Based Systems*, 13(2-3):53–59, 2000.
26. H. Tang, S. Tan, and X. Cheng. A survey on sentiment detection of reviews. *Expert Systems with Applications*, 36(7):10760–10773, 2009.
27. C. J. van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 1979.
28. J. Wiebe and E. Riloff. Creating subjective and objective sentence classifiers from unannotated texts. In *Proceedings of the 6th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing '05)*, pages 486–497, Mexico City, Mexico, 2005.
29. R. T. A. Wietsma and F. Ricci. Product reviews in mobile decision aid systems. In *Pervasive Mobile Interaction Devices (PERMID '05)*, pages 15–18, Munich, Germany, 2005.
30. Z. Zhang and B. Varadarajan. Utility scoring of product reviews. In *Proceedings of the 15th ACM international conference on Information and knowledge management (CIKM '06)*, pages 51–57, New York, NY, USA, 2006. ACM.

INTELLIGENT AGENTS

Agent Argumentation with Opinions and Advice

John Debenham & Carles Sierra

Abstract In argumentation-based negotiation the rhetorical illocutionary particles *Appeals*, *Rewards* and *Threats* have implications for the players that extend beyond a single negotiation and are concerned with building (business) relationships. This paper extends an agent's relationship-building argumentative repertoire with *Opinions* and *Advice*. A framework is described that enables agents to model their relationships and to use argumentative dialogue strategically both to achieve good negotiation outcomes and to build and sustain valuable relationships.

1 Introduction

The term *argumentation-based negotiation* has various meanings in multiagent systems [11]. *Classical argumentation* is the generation of arguments, usually as logical proofs, for and against a given course of action that support decision making processes. *Dialectical argumentation* is concerned with the argumentative process, and procedures by which argumentative dialogues are conducted. *Rhetorical argumentation* uses rhetorical illocutionary particles with the intention of modifying the beliefs of the listener. This paper is concerned with rhetorical argumentation.

Rhetorical argumentative dialogues have been traditionally organised around the rhetorical illocutionary particles *Offer*, *Accept* and *Reject* with the addition of particles such as *Appeals*, *Rewards* and *Threats* [17]. This form of argumentation has implications for the players that extend beyond a single negotiation and is concerned with building (business) relationships. When we reward or threaten we refer

John Debenham
QCIS, UTS, Broadway, NSW 2007, Australia, e-mail: debenham@it.uts.edu.au

Carles Sierra
IIIA, CSIC, Campus UAB, 08193 Bellaterra, Catalonia, Spain e-mail: sierra@iiia.csic.es

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_2,

to a future instant of time where the reward or threat will be effective, its scope goes beyond the current negotiation round.

This paper discusses the use of opinions and advice as rhetorical particles with the intention of building relationships. The intuition is that if an agent believes that another agent gives reliable and pertinent information, opinions and advice then this will strengthen their relationship. It is generally accepted that human agents rely heavily on their relationships in order to conduct business [19] [12]. An agent's *relationships* is a model of its interactions with other agents that enables it to exhibit a strategic social sense. The concept of trust [13] is one component of a relatlonship model — but there is far more to relationships than trust.

We will understand argumentation in this paper as an information exchange process between agents. Every illocution that an agent utters gives away valuable information. To evaluate each illocution exchanged we have built an information-based agent architecture on information theory. Information-based agents [15] have embedded tools from information theory that enable them to measure and manage strategic information — this makes them well-suited to measuring and managing the development of relationships through the exchange of rhetoric particles. The way in which this is achieved is described in detail in Section 6. As we will see, each agent has a world model that contains representations of every aspect of the world (including other agents) that the agent is interested in, as well as its strength of belief in the validity of the representation. When an information-based agent receives a rhetoric particle it updates its world model and calculates the *value* of that particle as information gain on the world model. Each particle is classified by identifying the components of the relationship model that it is relevant to, and its value is then used to update the strength of those components.

Section 2 describes the rhetoric particles: *informs*, *opinions*, and *advice*. The characteristics of relationships between agents are described and formalised in the LOGIC framework in Section 3. The formal model of relationships contains four component models that are all described in Section 4. Two further models, for trust and integrity, are presented in Section 5. The work then comes together in a discussion of strategies in Section 6, and Section 7 concludes.

2 Opinions and Advice

In this Section we describe the two rhetoric particles: `opinion` and `advise`:

- An `opinion` communicative act is a speaker's evaluation of a particular aspect of a thing in context, where the *context* is the set of all things that the thing is being, explicitly or implicitly, evaluated with or against.
- An `advise` communicative act is a speaker's evaluation of a particular aspect of a thing in the context of the speaker's beliefs of the listener's context. It is a directive in Searle's classification of speech acts.

Together with `inform`, these two acts are, in a sense, of increasing complexity. An `inform` communicates a verifiable fact, an `opinion` communicates the speaker's personal evaluation of something, and an `advice` communicates the speaker's opinion on what the listener's future states or actions should be.

We use the standard FIPA notation [1]. The rational effect (in the sense of the FIPA semantics) of these three particles is taken in two senses. First, the immediate effect that they will have on the listener's state or actions, and second, the effect that an act has on the relationship when the integrity of the information communicated has been verified. For the `inform` communicative act , agent i informs agent j of proposition p. The dual rational effects are:

$<i,$ `inform(`$j,\ p$ `)>`
`RE1:` $\mathrm{B}_j p$
`RE2: Done(<`$j,$ `eval(`$p,\ x$`)>,`ϕ`)`

where `RE1` is as in FIPA standard, and $<j,$ `eval(`$p,\ x$`)>` is the action of agent j rating the integrity of proposition p as x, and proposition ϕ is true when the integrity of p is known to j. The evaluation is performed *ex post* at a time when opportunities to use the contents of the `inform` are well understood. It is over a fuzzy scale, `eval` $\in [0,1]$, that must contain 0 (meaning "is of absolutely no use"), and must contain 1 (meaning "valued most highly").

The `opinion` and `advise` communicative acts are not part of the FIFA specification and are now defined using the FIPA notation.

The representation of the `opinion` communicative act contains:

- the thing that is the *subject* of the opinion,
- the *aspect*, or attribute, of the thing that is being evaluated,
- a distribution over some evaluation space representing the *rating* of the aspect of the thing in the context, and
- optionally the *context* in which the evaluation is made, and a *reason* supporting the opinion.

An `opinion` action indicates that the speaker:

- believes he knows that the listener holds a particular intention,
- believes his opinion of a thing is related to the listener's intention, and is more accurate than the listener's opinion of it

In the following, the speaker, i, informs the listener, j that his rating of an aspect, s, of a thing, t, is e in (the optional) context c for the (optional) reason, r. The two rational effects following represent the dual motives for uttering the illocution:

$<i,$ `opinion(`j,s,t,e`[,`c,r`])>`
`FP:` B_j `Rates(`j,s,t,e'`[,`c,r`])` $[\wedge \mathrm{B}_i \mathrm{I}_j c \wedge \mathrm{B}_i r] \wedge$
$\mathrm{B}_i \mathrm{I}_j$ `Done(<`$j,$ `eval(`s,t,e,x`[,`c,r`])>,`ϕ`)`
`RE1:` B_j `Rates(`j,s,t,e''`[,`c,r`])` $\wedge |e - e''| < |e - e'|$
`RE2: Done(<`$j,$ `eval(`s,t,e,x`[,`c,r`])>,`ϕ`)`

That is, i believes that as a result of expressing an opinion about t, j's rating of t is now closer to i's rating that it was prior to the opinion being uttered, where some suitable distance measure between distributions is assumed, and $\texttt{eval}(s,t,e,x[,c,r])$ is the action of evaluating the rating e in context, and ϕ is true when the evaluation is performed.

An `advise` *action* indicates that the speaker:

- believes he knows that the listener holds a particular intention,
- believes his knowledge of facts concerning the listener's intention is better than the listener's knowledge of them,
- intends the listener to believe that the advised action is in the listener's interests, and
- believes that the listener may act otherwise.

In the following, the speaker, i, advises the listener, j, that the speaker believes the listener should perform some action, a, if the listener's context includes the intention to achieve a goal, c. The two feasibility preconditions are alternative representations of i's beliefs of the superiority of his knowledge, and the two rational effects represent the dual motives for uttering the illocution:

$$
\begin{array}{ll}
\multicolumn{2}{l}{\texttt{<}i\texttt{, advise(}j\texttt{,}a\texttt{,}c\texttt{)>}} \\
\texttt{FP:} & \texttt{B}_i\texttt{I}_j c \wedge \texttt{B}_i(\ \texttt{W}_i(c) \rightarrow \texttt{W}_{j\backslash i}(c)\ ) \wedge \neg \texttt{B}_i\texttt{I}_j\ \texttt{Done(<}j\texttt{,}a\texttt{>)} \wedge \\
 & \texttt{B}_i\texttt{I}_j\ \texttt{Done(<}j\texttt{, eval(}a\texttt{,}c\texttt{,}x\texttt{)>,}\phi) \\
\texttt{or:} & \texttt{B}_i\texttt{I}_j c \wedge \texttt{B}_i(\mathbb{H}(\texttt{W}_i(c)) < \mathbb{H}(\texttt{W}_{j\backslash i}(c))) \wedge \neg \texttt{B}_i\texttt{I}_j\ \texttt{Done(}a\texttt{)} \wedge \\
 & \texttt{B}_i\texttt{I}_j\ \texttt{Done(<}j\texttt{, eval(}a\texttt{,}c\texttt{,}x\texttt{)>,}\phi) \\
\texttt{RE1:} & \texttt{Done(<}j\texttt{,}a\texttt{>)} \\
\texttt{RE2:} & \texttt{Done(<}j\texttt{, eval(}a\texttt{,}c\texttt{,}x\texttt{)>,}\phi)
\end{array}
$$

where:
$\texttt{eval}(a,c,x)$ is the action of evaluating action a as x in context c, as above
$\texttt{W}_i(c)$ denotes all of i's beliefs concerning c — i.e. that part of i's world model
$\texttt{W}_{j\backslash i}(c)$ denotes i's beliefs concerning all of j's beliefs concerning c
$\texttt{W}_i(c) \rightarrow \texttt{W}_{j\backslash i}(c)$ denotes that everything in $\texttt{W}_{j\backslash i}(c)$ can be derived from a subset of $\texttt{W}_i(c)$
$\mathbb{H}(S)$ denotes the overall uncertainty of the set of beliefs S — possibly as entropy

3 Relationships

A *relationship* between two agents is somehow encapsulated in their *history* that is a complete record of their interactions. This potentially large amount of information is usually summarised by agents into various models. For example, the majority of agents construct a world model and a trust model [3]. There is evidence from psychological studies that humans seek a *balance* in their negotiation relationships. The classical view [2] is that people perceive resource allocations as being distributively

fair (i.e. well balanced) if they are proportional to inputs or contributions (i.e. equitable). In the case of partners there is some evidence [4] that the allocations of goods and burdens (i.e. positive and negative utilities) are perceived as fair, or in balance, based on equity for burdens and equality for goods.

The LOGIC illocutionary framework for classifying argumentative interactions was first described in [16] where it was used to help agents to prepare for a negotiation in the *prelude stage* of an interaction. The work in this paper generalises that framework and uses it to define one of the two dimensions of the relationship model described in Section 4; the second dimension is provided by the structure of the ontology as specified by a partial order $\leq$ defined by the is-a hierarchy, and a distance measure between concepts such as Equation 1. The five LOGIC categories for information are quite general:

- Legitimacy contains *information* that may be part of, relevant to or in justification of contracts that have been signed.
- Options contains information about *contracts* that an agent may be prepared to sign.
- Goals contains information about the *objectives* of the agents.
- Independence contains information about the agent's *outside options* — i.e. the set of agents that are capable of satisfying each of the agent's needs.
- Commitments contains information about the *commitments* that an agent has.

and are used here to categorise all incoming communication that feeds into the agent's relationship model. As we will see this categorisation is not a one-to-one mapping and some illocutions fall into multiple categories. These categories are designed to provide a model of the agents' information that is relevant to their relationships, and are *not* intended to be a universal categorising framework for all utterances.

This paper is written from the point of view of an agent α is in a *multiagent system* with a finite number of other agents $\mathscr{B} = \{\beta_1, \beta_2, \dots\}$, and a finite number of *information providing agents* $\Theta = \{\theta_1, \theta_2, \dots\}$ that provide the *context* for all events in the system — Θ^t denotes the state of these agents at time t. α observes the actions of another agent β in the context Θ^t. The only thing that α 'knows for certain' is its *history* of past communication that is retains in the repository $\mathscr{H}_\alpha^t$. Each *utterance* in the history contains: an illocutionary statement, the sending agent, the receiving agent, the time that the utterance was sent or received.

Observations are of little value unless they can be verified. α may not posses a comprehensive range of reliable sensory input devices. Sensory inadequacy is dealt with invoking an *institution agent*, ξ, that truthfully, accurately and promptly reports what it sees.

All communication is recorded in α's history $\mathscr{H}_\alpha^t$ that in time may contain a large amount of data. The majority of agent architectures include models that summarise the contents of $\mathscr{H}^t$; for example, a *world model* and a *trust model*. In this paper we describe two models, a *relationship model* and an *integrity model* that are specifically designed to assist an agent to manage information asymmetries. To build the relationship model we will use the LOGIC framework to cat-

egorise the information in utterances received. That is, α requires a categorising function $v : U \to \mathscr{P}(\{\mathsf{L},\mathsf{O},\mathsf{G},\mathsf{I},\mathsf{C}\})$ where U is the set of utterances. The power set, $\mathscr{P}(\{\mathsf{L},\mathsf{O},\mathsf{G},\mathsf{I},\mathsf{C}\}$, is required as some utterances belong to multiple categories. For example, "I will not pay more for wine than the price that John charges" is categorised as both Option and Independence.

We assume an ontology that includes a (minimum) repertoire of elements: a set of *concepts* (e.g. quantity, quality, material) organised in a is-a hierarchy (e.g. platypus is a mammal, australian-dollar is a currency), and a set of relations over these concepts (e.g. price(beer,AUD)).[1]

We model ontologies following an algebraic approach [8]. An ontology is a tuple $\mathscr{O} = (C, R, \leq, \sigma)$ where:

1. C is a finite set of concept symbols (including basic data types);
2. R is a finite set of relation symbols;
3. $\leq$ is a reflexive, transitive and anti-symmetric relation on C (a partial order)
4. $\sigma : R \to C^+$ is the function assigning to each relation symbol its arity

where $\leq$ is a traditional *is-a* hierarchy, and R contains relations between the concepts in the hierarchy.

The concepts within an ontology are closer, semantically speaking, depending on how far away they are in the structure defined by the $\leq$ relation. Semantic distance plays a fundamental role in strategies for information-based agency. A measure [9] bases the *semantic similarity* between two concepts on the path length induced by $\leq$ (more distance in the $\leq$ graph means less semantic similarity), and the *depth* of the subsumer concept (common ancestor) in the shortest path between the two concepts (the deeper in the hierarchy, the closer the meaning of the concepts). Semantic similarity could then be defined as:

$$\mathrm{Sim}(c, c') = e^{-\kappa_1 l} \cdot \frac{e^{\kappa_2 h} - e^{-\kappa_2 h}}{e^{\kappa_2 h} + e^{-\kappa_2 h}} \tag{1}$$

where l is the length (i.e. number of hops) of the shortest path between the concepts, h is the depth of the deepest concept subsuming both concepts, and κ_1 and κ_2 are parameters scaling the contribution of shortest path length and depth respectively.

4 The Relationship Model $\mathscr{R}^t_{\alpha\beta}$

This Section describes how an agent's relationships are modelled using both the LOGIC framework (described in Section 3) and the structure of the ontology (described in Section 2). The relationship model is used in Section 6 to manage the argumentative discourse between two agents. Two models are described in Section 5

[1] Usually, a set of axioms defined over the concepts and relations is also required. We will omit this here.

that are used in Section 6 to select which agent to interact with in the context of a particular need.

All of α's models are summaries of its history $\mathscr{H}_\alpha^t$. The *relationship model* that α has of β consists of four component models. First, α's *intimacy model* of β's private information describes *how much* α knows about β, $I_{\alpha\beta}^t$ — this information will have been extracted from the dialogue including `inform`, `opinion` and `advise` utterances. Second, α's *reliability model* of *how reliable* the information summarised in $I_{\alpha\beta}^t$ is, $R_{\alpha\beta}^t$. Third, α's *reflection model* of β's model of α's private information, $J_{\alpha\beta}^t$. Fourth, a *balance model*, $B_{\alpha\beta}^t$, that measures the difference in the rate of growth of $I_{\alpha\beta}^t$ and $J_{\alpha\beta}^t$. Abusing notation we denote this by: $\frac{d}{dt}I_{\alpha\beta}^t$ and $\frac{d}{dt}J_{\alpha\beta}^t$ across the structure $\{\mathsf{L},\mathsf{O},\mathsf{G},\mathsf{I},\mathsf{C}\} \times \mathscr{O}$.

The remainder of this section details how these four component models are calculated. In addition to the models described in this Section, α is assumed to have a world model, $\mathscr{M}^t$, that represents everything in its world that it is interested in. The procedure for updating the world model relies on estimates of the reliability of all incoming utterances. $R_{\alpha\beta}^t$ is used for this purpose, and is used both to support the update process for $\mathscr{M}^t$ and to estimate the reliability of $I_{\alpha\beta}^t$. The description given employs the machinery to update the world model in our information-based agents [15]. However it can be adapted to the machinery used by any agent that represents uncertainty in its world model using probability distributions, that is: $\mathscr{M}^t = \{X_i\}_i$ where X_i are random variables. In addition to the world model and the models described in this paper an agent may construct other models such as an *honour model* [14].

Utterances are represented in the world model $\mathscr{M}_\alpha^t$ as probability distributions, (X_i), in first-order probabilistic logic $\mathscr{L}$. Representing an utterance in the world model requires its semantics. Semantics of utterances are specified as constraints on distributions in the world model. For example, in a simple multi-issue contract negotiation α may estimate $\mathbb{P}^t(\mathrm{acc}(\beta,\alpha,\delta))$, the probability that β would accept contract δ, by observing β's responses. The distribution $\mathbb{P}^t(\mathrm{acc}(\beta,\alpha,\delta)) \in \mathscr{M}_\alpha^t$ is classified as an Option in LOGIC. Using shorthand notation, if β sends the message $\mathrm{Offer}(\delta_1)$ then α derives the constraint: $K_{\mathrm{acc}(\beta,\alpha,\delta)}(\mathrm{Offer}(\delta_1)) = \{\mathbb{P}^t(\mathrm{acc}(\beta,\alpha,\delta_1)) = 1\}$, and if this is a counter offer to a former offer of α's, δ_0, then: $K_{\mathrm{acc}(\beta,\alpha,\delta)}(\mathrm{Offer}(\delta_1)) = \{\mathbb{P}^t(\mathrm{acc}(\beta,\alpha,\delta_0)) = 0\}$.[2]

Updating $\mathscr{M}_\alpha^t$ is complicated if the reliability of utterances received is taken into account — it would certainly be foolish for α to believe that every utterance received from β was correct — whereas all utterances received from the institution agent ξ *are* assumed to be correct. The procedure for doing this, and for attaching reliability estimates to utterances is described below.

The idea of intimacy and balance is that intimacy summarises the degree of closeness, and *balance* is degree of fairness. Informally, *intimacy* measures how

[2] In the not-atypical special case of multi-issue bargaining where the agents' preferences over the individual issues *only* are known and are complementary to each other's, maximum entropy reasoning can be applied to estimate the probability that any multi-issue offer will be acceptable to β by enumerating the possible worlds that represent β's "limit of acceptability" [15].

much one agent knows about another agent's private information, and *balance* measures the extent to which information revelation process between the agents is 'fair'. The *intimacy* and *balance* models are structured using the LOGIC illocutionary framework and the ontology $\mathscr{O}$[3]. For example, an utterance meaning that agent β accepts agent α's previously offered deal δ is classified as an Option, and $<\alpha, \texttt{inform}(\beta, info)>$ meaning that agent β informs α about *info* and commits to the truth of it is classified as Legitimacy.

4.1 The Intimacy Model: $I^t_{\alpha\beta}$

The *intimacy* of α's relationship with β, $I^t_{\alpha\beta}$, models how much α knows about β's private information and is represented as real numeric values over $\{\mathsf{L},\mathsf{O},\mathsf{G},\mathsf{I},\mathsf{C}\} \times \mathscr{O}$. Suppose α receives an utterance u from β and that the LOGIC category $f \in v(u)$, where v is the categorising function described in Section 3. For any concept $c \in \mathscr{O}$, we extend the definition of Sim by defining $\text{Sim}(u,c) = \max_{c' \in u} \text{Sim}(c',c)$ where Sim is a semantic distance function such as that described in Equation 1. Denote the value of $I^t_{\alpha\beta}$ in position $(f,c) \in \{\mathsf{L},\mathsf{O},\mathsf{G},\mathsf{I},\mathsf{C}\} \times \mathscr{O}$ by $I^t_{\alpha\beta(f,c)}$ then:

$$I^t_{\alpha\beta(f,c)} = \begin{cases} \rho \times I^{t-1}_{\alpha\beta(f,c)} + (1-\rho) \times \mathbb{I}^t(u) \times \text{Sim}(u,c) & \text{if } u \text{ received,} \\ \mu \times I^{t-1}_{\alpha\beta(f,c)} & \text{otherwise.} \end{cases} \quad (2)$$

for any c, where $\mu < 1$ is the decay rate, ρ is the learning rate, and $\mathbb{I}^t(u)$ is Shannon information gain as given by Equation 7 that is described below. The method for estimating $\mathbb{I}^t(u)$ takes account of the reliability of u. The decay rate μ is a constant just less than 1 ensures the decay of $I^t_{\alpha\beta}$ towards a zero state if no utterances are received. α's estimate of β's intimacy on α, $J^t_{\alpha\beta}$, is constructed similarly by assuming that β's reasoning apparatus mirrors α's.

Equation 2 above requires an estimate of the information gain in an utterance, $\mathbb{I}^t(u)$. The calculation is fairly technical but as it is part of the procedure for updating the world model the marginal cost in building the relationship model is very low.

α's world model $\mathscr{M}^t_\alpha$ is a set of random variables, $\mathscr{M}^t = \{X_i, \ldots, X_n\}$ each representing an aspect of the world that α is interested in. In the absence of in-coming messages the integrity of $\mathscr{M}^t$ decays. α may have background knowledge concerning the expected integrity as $t \to \infty$. Such background knowledge is represented as a *decay limit distribution*. One possibility is to assume that the decay limit distribution has maximum entropy whilst being consistent with observations. Given a distribution, $\mathbb{P}(X_i)$, and a decay limit distribution $\mathbb{D}(X_i)$, $\mathbb{P}(X_i)$ decays by:

$$\mathbb{P}^{t+1}(X_i) = \Delta_i(\mathbb{D}(X_i), \mathbb{P}^t(X_i)) \quad (3)$$

[3] Only a subset of the ontology is required. The idea is simply to capture "How much has Carles told me about wine", or "how much do I know about Carles' commitments (possibly with other agents) concerning cheese".

where Δ_i is the *decay function* for the X_i satisfying the property that $\lim_{t\to\infty}\mathbb{P}^t(X_i) = \mathbb{D}(X_i)$. For example, Δ_i could be linear: $\mathbb{P}^{t+1}(X_i) = (1-\mu_i)\times\mathbb{D}(X_i) + \mu_i \times \mathbb{P}^t(X_i)$, where $\mu_i < 1$ is the decay rate for the i'th distribution. Either the decay function or the decay limit distribution could also be a function of time: Δ_i^t and $\mathbb{D}^t(X_i)$.

Suppose that α receives an utterance u from agent β at time t. This utterance could be an `inform`, an `opinion` or an `advise`. Suppose that this utterance's contents is qualified with probability z. α attaches an epistemic belief $R^t_{\alpha\beta}(u)$ to u — the reliability model $R^t_{\alpha\beta}$ is described below in Section 4.2. The semantics of utterance u is given by specifying constraints on those random variables in the world model that the receipt of u will effect. For $X_i \in \mathscr{M}^t$ we denote the constraint on X_i due to the receipt of u as $K_{X_i}(u)$ that are called *update functions*.

Given a prior distribution $\mathbf{p}_i = \mathbb{P}^t(X_i)$ let $\mathbf{p}_{i(u)}$ be the distribution with minimum relative entropy[4] with respect to $\mathbf{p}_i$: $\mathbf{p}_{i(u)} = \arg\min_{\mathbf{r}} \sum_j r_j \log \frac{r_j}{p_j}$ that satisfies the constraints $K_{X_i}(u)$. Then let $\mathbf{q}_{i(u)}$ be the distribution:

$$\mathbf{q}_{i(u)} = \begin{cases} R^t_{\alpha\beta}(u) \times \mathbf{p}_{i(u)} + (1 - R^t_{\alpha\beta}(u)) \times \mathbf{p}_i & \text{if } R^t_{\alpha\beta}(u) > 0.5, \\ \mathbf{p}_i & \text{otherwise.} \end{cases} \tag{4}$$

where $R^t_{\alpha\beta}(u)$ is determined by the reliability model below. The condition $R^t_{\alpha\beta}(u) > 0.5$ prevents information with an expected evaluation less than the ambivalence point (i.e. 0.5 as discussed in Section 4.2) from entering the process for updating $\mathscr{M}^t$. For example, $R^t_{\alpha\beta}(u) = 0$ means that u is certainly of no value. Then let:

$$\mathbb{P}^t(X_{i(u)}) = \begin{cases} \mathbf{q}_{i(u)} & \text{if } \mathbf{q}_{i(u)} \text{ is "more interesting" than } \mathbf{p}_i \\ \mathbf{p}_i & \text{otherwise} \end{cases} \tag{5}$$

A general measure of whether $\mathbf{q}_{i(u)}$ is *more interesting* than $\mathbf{p}_i$ is: $\mathbb{K}(\mathbf{q}_{i(u)} \| \mathbb{D}(X_i)) > \mathbb{K}(\mathbf{p}_i \| \mathbb{D}(X_i))$, where $\mathbb{K}(\mathbf{x}\|\mathbf{y}) = \sum_j x_j \ln \frac{x_j}{y_j}$ is the Kullback-Leibler distance between two probability distributions $\mathbf{x}$ and $\mathbf{y}$.

Finally merging Equation 5 and Equation 3 we obtain the method for updating a distribution X_i on receipt of a message u:

$$\mathbb{P}^{t+1}(X_i) = \Delta_i(\mathbb{D}(X_i), \mathbb{P}^t(X_{i(u)})) \tag{6}$$

This procedure deals with integrity decay, and with two probabilities: first, the probability z in the utterance u, and second the reliability $R^t_{\alpha\beta}(u)$ that α attached to u.

[4] Given a probability distribution $\mathbf{q}$, the *minimum relative entropy distribution* $\mathbf{p} = (p_1, \ldots, p_I)$ subject to a set of n linear constraints $\mathbf{g} = \{g_j(\mathbf{p}) = \mathbf{a_j} \cdot \mathbf{p} - c_j = 0\}, j = 1, \ldots, n$ (that must include the constraint $\sum_i p_i - 1 = 0$) is: $\mathbf{p} = \text{MRE}(\mathbf{q}, \mathbf{g}) = \arg\min_{\mathbf{r}} \sum_j r_j \log \frac{r_j}{q_j}$. This may be calculated by introducing Lagrange multipliers λ: $L(\mathbf{p}, \lambda) = \sum_j p_j \log \frac{p_j}{q_j} + \lambda \cdot \mathbf{g}$. Minimising L, $\{\frac{\partial L}{\partial \lambda_j} = g_j(\mathbf{p}) = 0\}, j = 1, \ldots, n$ is the set of given constraints $\mathbf{g}$, and a solution to $\frac{\partial L}{\partial p_i} = 0, i = 1, \ldots, I$ leads eventually to $\mathbf{p}$. Entropy-based inference is a form of Bayesian inference that is convenient when the data is sparse [5] and encapsulates common-sense reasoning [10].

The Shannon information gain in X_i is: $\mathbb{I}^t X_i = \mathbb{H}^t(X_i) - \mathbb{H}^{t-1}(X_i)$, and if the distributions in $\mathscr{M}^t$ are independent then the Shannon information gain for $\mathscr{M}^t$ following the receipt of utterance u is:

$$\mathbb{I}^t(u) = \sum_{X_i} \mathbb{I}^t X_i \tag{7}$$

4.2 The Reliability Model: $R^t_{\alpha\beta}$

Equation 4 above requires an estimate of the reliability of an utterance, $R^t_{\alpha\beta}(u)$, which is detailed in this Section. The reliability model is constructed by observing the difference between β's utterance u at time t and its subsequent evaluation[5] at time t'. This means that for β, building a strong $R^t_{\alpha\beta}$ will be a slow process. This is consistent with the observation that business relationships between human agents tend to build gradually over time.

We now consider how the estimates $R^t_{\alpha\beta(f,c)}$ develop in time. At each time step:

$$R^t_{\alpha\beta(f,c)} = \mu \times R^{t-1}_{\alpha\beta(f,c)} + (1-\mu) \times 0.5$$

representing the decay of the reliability towards the maximum entropy, or ambivalence point, value. Now suppose that u is received from agent β at some time and is evaluated, possibly with the assistance of the institution agent, ξ, at some later time t as $\texttt{eval}(u)$ as described in Section 2. This evaluation is on a fuzzy scale in $[0,1]$ that contains 0 and 1, i.e. $\texttt{eval}(u) \in [0,1]$. Suppose that the LOGIC category $f \in v(u)$, where v is the categorising function described in Section 3. For any category c, let $r = R^{t-1}_{\alpha\beta(f,c)}$ and:

$$\begin{aligned} e &= (\rho \times 0.5) + (1-\rho) \times \texttt{eval}(u) \\ e' &= e \times \mathrm{Sim}(u,c) \\ e'' &= (\mathrm{Sim}(u,c) \times (e-1)) + 1 \end{aligned}$$

where ρ is the learning rate, then $R^t_{\alpha\beta(f,c)} = g(r,e',e'')$ where:

$$g(r,e',e'') = \begin{cases} \mathrm{comb}(r,e') & \text{if } e > 0.5 \text{ and } e' > 0.5, \\ \mathrm{comb}(r,e'') & \text{if } e < 0.5 \text{ and } e'' < 0.5, \\ r & \text{otherwise.} \end{cases} \tag{8}$$

where $\mathrm{comb}(x,y) = \frac{x \times y}{(x \times y) + (1-x) \times (1-y)}$ is the combination of independent probabilities x and y. The assumption of independence is rather radical and the moderation of $\texttt{eval}(u)$ to e using the learning rate ρ is intended to compensate for this. The con-

[5] Evaluation is meant in the sense of the `eval` functions that are part of the rational effect expressions in Section 2.

ditions in Equation 8 ensures that the update is only applied when Sim is reasonably large. When $\mathrm{Sim} = 1$, $e' = e'' = e$. Those conditions limit the update to those values of e' and e'' that are "on the same side of" 0.5 as e.

5 Trust and Integrity

We now describe two measures that are attached to *complete dialogues* that are used in Section 6 to assist with the selection of negotiation partners for a particular need. The first of these is *trust* that measures the difference been commitments made during a dialogue and the eventual enactment of those commitments. The second is *integrity* that measures the difference between expectation and evaluation of the dialogue — the integrity measure is aggregated from values of the `eval` function.

The estimation of trust and integrity can be interpreted as a pattern mining exercise from the information in $\mathscr{C}^t_{\alpha\beta}$ to find the 'best' hypothesis that describes $\mathscr{C}^t_{\alpha\beta}$, where $\mathscr{C}^t_{\alpha\beta} \subset \mathscr{H}^t_{\alpha\beta}$ contains those utterances that contain evaluations of enactments, for trust, and of consumption, for integrity. One neat way to perform this induction is the *minimum description length principle* [7] that is founded on the minimisation of the cost of communicating a body of knowledge from one agent to another that thus has a fundamental affinity with distributed autonomous systems:

$$\mathscr{I}^t_{\alpha\beta} \triangleq \arg\min_M (\mathbb{L}(M) + \mathbb{L}(\mathscr{C}^t_{\alpha\beta} \mid M)) \tag{9}$$

where $\mathbb{L}(M)$ is the length of the shortest encoding of M, and $\mathbb{L}(\mathscr{C}^t_{\alpha\beta} \mid M)$ is the length of the shortest encoding of $\mathscr{C}^t_{\alpha\beta}$ given M. This definition is as neat as it is computationally expensive — it divides $\mathscr{C}^t_{\alpha\beta}$ into that which may be generalised and that which may not.

The definition of $\mathscr{I}^t_{\alpha\beta}$ in Equation 9 appears problematic for three reasons. First, if M can be any Turing computable model the definition is not computable, second a single language is required for representing M, and third the meaning of 'the length of the shortest encoding' is not clear. The second and third reason have been resolved [7]. The first, computability problem can be solved by restricting the models to some specific class. If the models are restricted to Bayesian decision graphs over finite spaces then Equation 9 is computable [18].

Equation 9 does not take time into account. To allow for varying strength of observations with time we construct instead $\mathscr{C}^{*t}_{\alpha\beta}$ that is the same as $\mathscr{C}^t_{\alpha\beta}$ except each evaluation, x, is replaced by a random variable X over evaluation space. These probability distributions are constructed by: $\lambda \times X + (1-\lambda) \times D_X$ where D_X is the *decay limit distribution*[6] for X — and X is a distribution with a '1' indicating the position of the evaluation and 0's elsewhere. Despite its elegance, Equation 9 is computationally expensive. [15] describes a computationally friendly method for evaluating trust that may also be used for integrity.

[6] If the decay limit distribution is unknown we use a maximum entropy distribution.

6 'Relationship-aware' Argumentation Strategies

Given a need ν in context Θ^t one way for agent α to select an interaction partner, β, on the basis of their past behaviour, is by reference to the trust model and the integrity model. Suppose that α uses model $M^t_{\alpha\beta}$, then the integrity of that model will decay in time, $M^t_{\alpha\beta} \to D$, by Equation 3 or similar, that is, the uncertainty $\mathbb{H}(M^t_{\alpha\beta})$ will increase in time, until the model is refreshed with new observations. So the rate of refreshment with new observations needs to be such that the uncertainty, $\mathbb{H}(M^t_{\alpha\beta})$, generally decreases in time. Given a need ν the set of partners that α considers are called the *pool* for ν. For each potential partner, β, we assume that α is able to estimate, $\mathbb{P}^t(\beta \gg)|\nu$, the probability that a negotiation with β to satisfy ν will lead to a better outcome than with any other in the pool. Then select a partner using the stochastic strategy: $\mathbb{P}^t(\text{Select } \beta_i) = \mathbb{P}^t(\beta_i \gg)|\nu$.

For each agent in the pool α has a view on the desired form of the relationship model, particularly the intimacy model, $I^t_{\alpha\beta}$ — that is the model that he would realistically wish to have. This is the *relationship target* $T^t_{\alpha\beta}$ for agent β. Using the {L,O,G,I,C} structure the target is expressed as a target for the pair of intimacy models in Section 4.1: $T^t_{\alpha\beta(f,c)} = \left(TI^t_{\alpha\beta(f,c)}, TJ^t_{\alpha\beta(f,c)}\right)$ where $TI_{\alpha\beta}$ and $TJ_{\alpha\beta}$ are respectively the targets for $I_{\alpha\beta}$ and $J_{\alpha\beta}$.

Having selected the interaction partner, and having set the relationship target, α now manages the interaction itself. α has a model of its intimacy with β, $I^t_{\alpha\beta(f,n)}$ where $\nu \in n \in \mathscr{O}$, and its target for β, $T^t_{\alpha\beta(f,n)}$. When the interaction with β is complete, the intimacy model, $I^{t'}_{\alpha\beta(f,n)}$ will have changed. Before the interaction commences, α may desire that $I^{t'}_{\alpha\beta(f,n)}$ will have changed "in the direction of" $T^t_{\alpha\beta(f,n)}$. This is formalised as the *negotiation target*, $N^t_{\alpha\beta(f,c)} = \left(NI^t_{\alpha\beta(f,c)}, NJ^t_{\alpha\beta(f,c)}\right)$, that is α's aspirations at time t for intimacy at time t'. Given the uncertainty in behaviour in any negotiation, the negotiation target is an approximate indication only of what should be achieved.

Any utterance that an agent makes gives away information if the receiving agent revises its world model as a result of receiving the utterance. In single-issue `offer`, `accept` and `reject` negotiation the *equitable information revelation* strategy is: α responds to β's offer with an offer o that gives β equivalent information gain as α has observed provided that o is acceptable to α[7]. Formally, if α receives an offer o from β at time t then α will observe information gain $\mathbb{H}(\mathscr{M}^{t-1}) - \mathbb{H}(\mathscr{M}^t)$ and so responds with an offer o' which is such that: $\mathbb{H}(\mathscr{M}^t_\beta) - \mathbb{H}(\mathscr{M}^t_\beta \oplus o') \approx \mathbb{H}(\mathscr{M}^{t-1}) - \mathbb{H}(\mathscr{M}^t)$ as long as o' is acceptable to α. If the negotiation is single-issue then this strategy determines a unique offer and yields a sequence of alternating offer exchanges that is almost "classic market haggling".

[7] This assumes, not unreasonably, that α and β model each other's limit price with a random variable in their respective world models.

For multi-issue `offer`, `accept` and `reject` negotiation we assume that α estimates the probability that any proposed deal, δ, is acceptable, $\mathbb{P}^t(Acc_\alpha(\delta|\mathscr{H}^t_\alpha))$, that is accompanied by a threshold value τ meaning that if $\mathbb{P}^t(Acc_\alpha(\delta|\mathscr{H}^t_\alpha)) > \tau$ then δ is acceptable. We also assume that α estimates the probability that the deal will be acceptable to β, $\mathbb{P}^t(Acc_\beta(\delta|\mathscr{H}^t_\alpha))$; an estimate for this may be derived from the offers that β has both made and rejected using maximum entropy inference [14]. Given these two estimates then an analogue of the issue-tradeoffs strategy in [6] is for α to offer: $\delta^* = \arg\max_\delta\{\mathbb{P}^t(Acc_\beta(\delta|\mathscr{H}^t_\alpha)) \mid \mathbb{P}^t(Acc_\alpha(\delta|\mathscr{H}^t_\alpha)) > \tau\}$.

The issue-tradeoffs strategy described above does not take into account the expected information gain from making such a proposal: $\mathbb{H}(\mathscr{M}^t_\beta) - \mathbb{H}(\mathscr{M}^t_\beta \oplus \delta)$. The consideration of information gain adds an interesting dimension. Consider the set of deals of similar acceptability to β as δ^*: $\Delta = \{\delta \mid \mathbb{P}^t(Acc_\beta(\delta|\mathscr{H}^t_\alpha)) \approx \delta^* \wedge \mathbb{P}^t(Acc_\alpha(\delta|\mathscr{H}^t_\alpha)) > \tau\}$. Each $\delta \in \Delta$ are similarly acceptable to each agent but are of potentially different information gain to β: $\mathbb{H}(\mathscr{M}^t_\beta) - \mathbb{H}(\mathscr{M}^t_\beta \oplus \delta)$. α is now is a position to decide how to manage the revelation of information in the proposals it makes, and may decide to do so equitably or otherwise.

The term *tactics* is used to refer to the strategy that wraps a possibly empty proposal in argumentation to form a complete utterance. The equitable information revelation strategy extends without modification to argumentation across the full structure of {L,O,G,I,C}. If α receives an utterance u from β at time t then α responds with u' which is such that: $\mathbb{H}(\mathscr{M}^t_\beta) - \mathbb{H}(\mathscr{M}^t_\beta \oplus u') \approx \mathbb{H}(\mathscr{M}^{t-1}) - \mathbb{H}(\mathscr{M}^t)$ as long as any contractual commitment in u' is acceptable to α. The idea is that α uses the negotiation target as a guide to go above or below an equitable information revelation response.

The negotiation literature consistently advises that an agent's behaviour should not be predictable even in close, intimate relationships. This variation of behaviour is normally described as varying the negotiation *stance* that informally varies from "friendly guy" to "tough guy". The stance injects bounded random noise into the process, where the bound tightens as intimacy increases. For software agents, the role of stance is to prevent an observer from decrypting an agent's strategies.

7 Discussion

The prospect of automating the negotiation process in electronic business is a powerful motivation for research into robust negotiation strategies. A considerable effort is being made by game theorists to build strategies on a utilitarian basis. The work described in this paper is concerned with aspects of negotiation that are difficult, if not impossible, to capture within the utilitarian framework. Specifically the work is concerned with building relationships with the intention that they will provide agents with some degree of protection against the exploitation of information asymmetries in the marketplace. The strategic use of opinions and advice as argumentative illocutionary particles is one step on a long road to build reliable agents for business negotiation.

References

1. FIPA Communicative Act Library Specification. Tech. Rep. SC00037J, Foundation for Intelligent Physical Agents, Geneva, Switzerland (2002)
2. Adams, J.S.: Inequity in social exchange. In: L. Berkowitz (ed.) Advances in experimental social psychology, vol. 2. New York: Academic Press (1965)
3. Artz, D., Gil, Y.: A survey of trust in computer science and the semantic web. Web Semantics: Science, Services and Agents on the World Wide Web **5**(2), 58–71 (2007)
4. Bazerman, M.H., Loewenstein, G.F., White, S.B.: Reversal of preference in allocation decisions: judging an alternative versus choosing among alternatives. Administration Science Quarterly (37), 220–240 (1992)
5. Cheeseman, P., Stutz, J.: Bayesian Inference and Maximum Entropy Methods in Science and Engineering, chap. On The Relationship between Bayesian and Maximum Entropy Inference, pp. 445 – 461. American Institute of Physics, Melville, NY, USA (2004)
6. Faratin, P., Sierra, C., Jennings, N.: Using similarity criteria to make issue trade-offs in automated negotiation. Journal of Artificial Intelligence **142**(2), 205–237 (2003)
7. Grünwald, P.D.: The Minimum Description Length Principle. MIT Press, Cambridge, MA (2007)
8. Kalfoglou, Y., Schorlemmer, M.: IF-Map: An ontology-mapping method based on information-flow theory. In: S. Spaccapietra, S. March, K. Aberer (eds.) Journal on Data Semantics I, *Lecture Notes in Computer Science*, vol. 2800, pp. 98–127. Springer-Verlag: Heidelberg, Germany (2003)
9. Li, Y., Bandar, Z.A., McLean, D.: An approach for measuring semantic similarity between words using multiple information sources. IEEE Transactions on Knowledge and Data Engineering **15**(4), 871 – 882 (2003)
10. Paris, J.: Common sense and maximum entropy. Synthese **117**(1), 75 – 93 (1999)
11. Rahwan, I., Ramchurn, S., Jennings, N., McBurney, P., Parsons, S., Sonenberg, E.: Argumentation-based negotiation. Knowledge Engineering Review **18**(4), 343–375 (2003)
12. Rauyruena, P., Miller, K.E.: Relationship quality as a predictor of B2B customer loyalty. Journal of Business Research **60**(1), 21–31 (2007)
13. Sabater, J., Sierra, C.: Review on computational trust and reputation models. Artificial Intelligence Review **24**(1), 33–60 (2005)
14. Sierra, C., Debenham, J.: Trust and honour in information-based agency. In: P. Stone, G. Weiss (eds.) Proceedings Fifth International Conference on Autonomous Agents and Multi Agent Systems AAMAS-2006, pp. 1225 – 1232. ACM Press, New York, Hakodate, Japan (2006)
15. Sierra, C., Debenham, J.: Information-based agency. In: Proceedings of Twentieth International Joint Conference on Artificial Intelligence IJCAI-07, pp. 1513–1518. Hyderabad, India (2007)
16. Sierra, C., Debenham, J.: The LOGIC Negotiation Model. In: Proceedings Sixth International Conference on Autonomous Agents and Multi Agent Systems AAMAS-2007, pp. 1026–1033. Honolulu, Hawai'i (2007)
17. Sierra, C., Jennings, N., Noriega, P., Parsons, S.: Proceedings of the 4th International Workshop on Intelligent Agents IV, Agent Theories, Architectures, and Languages, chap. A Framework for Argumentation-Based Negotiation, pp. 177 – 192. Springer-Verlag, London, UK (1997)
18. Suzuki, J.: Learning bayesian belief networks based on the MDL principle: An efficient algorithm using the branch and bound technique. IEICE TRANSACTIONS on Information and Systems **E81-D**(12), 356–367 (1998)
19. Ulaga, W., Eggert, A.: Relationship value in business markets: The construct and its dimensions. Journal of Business To Business Marketing **12**(1), 73 – 99 (2005)

Graph-Based Norm Explanation

Madalina Croitoru and Nir Oren and Simon Miles and Michael Luck

Abstract Norms impose obligations, permissions and prohibitions on individual agents operating as part of an organisation. Typically, the purpose of such norms is to ensure that an organisation acts in some socially (or mutually) beneficial manner, possibly at the expense of individual agent utility. In this context, agents are *norm-aware* if they are able to reason about which norms are applicable to them, and to decide whether to comply with or ignore them. While much work has focused on the creation of norm-aware agents, much less has been concerned with aiding system designers in understanding the *effects* of norms on a system. The ability to understand such norm effects can aid the designer in avoiding incorrect norm specification, eliminating redundant norms and reducing normative conflict. In this paper, we address the problem of norm understanding by providing *explanations* as to why a norm is applicable, violated, or in some other state. We make use of conceptual graph based semantics to provide a graphical representation of the norms within a system. Given knowledge of the current and historical state of the system, such a representation allows for explanation of the state of norms, showing for example *why* they may have been activated or violated.

Madalina Croitoru
LIRMM, University Montpellier II, France, e-mail: croitoru@lirmm.fr

Nir Oren
Dept. of Computer Science, University of Aberdeen, UK e-mail: n.oren@abdn.ac.uk

Simon Miles
Dept. of Informatics, King's College London, UK e-mail: simon.miles@kcl.ac.uk

Michael Luck
Dept. of Informatics, King's College London, UK e-mail: michael.luck@kcl.ac.uk

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_3,

1 Introduction

Norm-aware agents make use of concepts such as obligations, permissions, and prohibitions, to represent and reason about socially imposed goals and capabilities. Such agents are able to decide whether to act in a manner consistent with norms, or whether to ignore them. Typically, norms are imposed on a set of agents in order to increase the overall utility of a system or society (often at the cost of individual utility)[9], or to reduce computational or communication overhead [4].

While a norm-aware agent is able to reason about which norms are applicable to it, or to another agent given a particular context, the problem of *explaining* why a norm is applicable, or violated, or in some other similar state, has not been investigated in depth. Yet the ability to provide such an explanation has multiple benefits. For example, system designers would be better able to understand the interactions between different norms, allowing them to avoid creating redundant norms [3], and to specify their norms more precisely. Conversely, users would be able to elicit a more intuitive understanding of the operation of a system by establishing the reasons why certain norms were assigned a particular status in response to system events.

Norms are typically specified within some knowledge-based system using a logic which, for non-technical users, is often difficult to understand. Such knowledge-based systems (KBS) are designed in order to represent knowledge (within a knowledge base) in a such a way that reasoning can be performed over it. In turn, a knowledge base is built on top of some set of *ontologies*. From an epistemological viewpoint, an ontology answers the question "what kinds of things exist in the application domain?" For our normative framework we consider computational ontologies, which provide a symbolic representation of classes of objects, called *concepts*, as well as the possible relationships between objects, called *relations* or *roles*. All other pieces of knowledge in the KBS are expressed by structures built with the ontology terms (concepts and relations). For a KBS to be usable, it is essential that the user understands and controls not only the knowledge base construction process, but also how results are obtained from the running system. It should be easy for this user not only to enter different pieces of knowledge and to understand their meaning but also to understand the results of the system and how the system computed these results. The last point, namely the ability to understand why the system gives a certain answer, is especially important since the user's computing expertise may vary. Given the difficulty non-specialists have in understanding formal textual explanations of the logical inference process, we propose a graphical norm representation, based on conceptual graphs [10], which have a sound and complete semantics with respect to a subset of first order logic [5]. The benefits of using graphs for representing knowledge stem from the following:

- First, graphs are simple mathematical objects (they only use elementary naive set theory notions such as elements, sets and relations) which have graphical representations (sets of points and lines connecting some pairs of points) and thus can be visualised.

- Second, graphs can be equipped with a logical semantics: the graph-based mechanisms they are provided with are sound and complete with respect to deduction in the assigned logic.

Our goal in this paper is to provide a graph-based semantics to the normative framework found in [8]. This normative framework was designed with a number of purposes in mind, namely to allow for the monitoring of the changing status of norms, and to support agent reasoning regarding norms. The semantics we describe allows one to graphically represent the changes in norms, and to determine their status using graph based operations such as projection. Thus, we are able to provide a visual explanation of certain aspects of normative reasoning.

2 Background

2.1 The Normative Model

Due to space constraints, we do not provide a complete formal description of the normative model. Instead, we describe the model by examining how it may be applied to a small example. Consider a situation where an agent *Alice* takes her car (an *Audi*) to a repair shop in order to be repaired. This repair shop provides a guarantee to its customers that their cars will be repaired within seven days.

The repair shop thus has an obligation upon it whenever a car arrives, to repair it within seven days. Clearly, once this obligation is fulfilled, it is lifted, and the repair shop no longer needs to repair the car. However, the obligation remains on the repair shop as long as the car is not repaired (even after seven days have passed).

Given this example, we observe that a norm may be defined in terms of five components. First, a norm has a *type*, for example, an obligation, or a permission. Second, a norm has an *activation condition*, identifying the situations in which the norm affects some agents. Third, a norm imposes some *normative condition* on the affected agent; if this normative condition does not hold, the norm is not being followed (i.e. in the case of our obligation, it is *violated*). Fourth, norms have a termination, or *expiration condition*, identifying the situations after which the norm no longer affects the agent. Finally, the norm must identify those agents to which it applies, known as the *norm targets*.

Note that the requirement on the repair shop to repair a car within seven days only obliges the repair shop to take action once a car actually arrives. Until then, the norm is an *abstract norm*. Then, when a customer brings in a car, the norm is instantiated, imposing a normative requirement upon the repair shop, and obliging it to repair the car within seven days. A single abstract norm can result in multiple *instantiated norms*; if two cars arrive at the repair shop, two instantiations of the abstract norm will occur.

More formally, we assume that the permissions and obligations represented by the norm refer to states and events in some environment, represented by some logical predicate language $\mathscr{L}$, such as first order logic. A norm is then a tuple of the form:

$$\begin{aligned}\langle &NormType,\\ &NormActivation,\\ &NormCondition,\\ &NormExpiration,\\ &NormTarget\rangle\end{aligned}$$

where

1. $NormType \in \{obligation, permission\}$
2. $NormActivation, NormCondition, NormExpiration$, and $NormTarget$ are all well formed formulae (*wff*) in $\mathscr{L}$.

Thus, for example, the following abstract norm represents the idea that a repair shop must repair a car within seven days of its arrival at the shop[1]:

$$\begin{aligned}\langle &obligation,\\ &arrivesAtRepairShop(X, Car, T_1),\\ &repaired(Car) \vee (currentTime(CurrentTime) \wedge\\ &before(CurrentTime, T_1 + 7days)),\\ &repaired(Car),\\ &repairShop(X)\rangle\end{aligned}$$

The predicate labels in this example refer to both events and states. This was done for ease of presentation; the use of a more complex underlying language would disambiguate these concepts, and provide us with a richer typology of temporal concepts.

In [8], a logical semantics for the instantiation and processing of norms represented using the tuple representation of norms is described. The tuple's attributes map directly onto the five components of a norm detailed above.

Temporal notions play a major role in this model of norms. A norm is instantiated at some time when the norm's activation conditions hold. The instantiated norm then persists, regardless of the valuation of the activation condition, until the norm's expiration conditions hold. Finally, we identify agents by constants of $\mathscr{L}$. Therefore, if we assume that some car car_1 arrives at Bob's repair shop at time 12, we would instantiate the abstract norm and obtain the following instantiated norm:

$$\langle obligation,$$

[1] Unless otherwise stated, we make use of Prolog notation within our logical formulae. More specifically, variables are written with an initial capital letter, while constants begin with a lowercase letter.

$$\begin{aligned} &arrivesAtRepairShop(bob, car_1, 12),\\ &repaired(car_1) \vee (currentTime(CurrentTime) \wedge\\ &before(CurrentTime, 19)),\\ &repaired(car_1),\\ &repairShop(bob)\rangle \end{aligned}$$

One issue we have not yet addressed is where, conceptually, norms are stored. It is clear that norms are not predicates (though they may be represented as such). We thus assume the existence of a separate *normative environment*, which is used to keep track of the abstract and instantiated norms within the system. Since norms may be instantiated and expire as time passes, the normative environment must, at each time point, identify which norms exist. One possible implementation of the normative environment is described in [8].

One of the main purposes of this normative model is to identify the changing status of norms over time. A norm's status may include the fact that it is instantiated or abstract, whether it is being complied with or violated, and whether it has expired. This status may be referred to by other norms. For example, a norm stating that "if a car is in the shop, and must be repaired within 7 days, and seven days have not yet passed, it is possible to request an extension for this repair work" could be written as follows (given that the norm above is labelled $n1$ and that the action of requesting a delay is written using the *requestDelay* predicate):

$$\begin{aligned} &\langle permission,\\ &active(n1) \wedge \neg violated(n1),\\ &requestDelay(X, Car),\\ &expired(n1) \vee violated(n1),\\ &repairShop(X)\rangle \end{aligned}$$

The $violated(n1)$ predicate makes use of the norm's status, and evaluates to true if and only if $n1$ is an instantiated obligation whose normative condition evaluates to false, and for which there is no permission that allows the negation of the normative condition. The $active(n1)$ predicate returns true if norm $n1$ is active, and the $expired(n1)$ predicate returns true if $n1$ has expired. These, and other such predicates are formally defined in [8].

As seen in the norm above, norms can often refer to other norms and the variables found within them (e.g. *Car* in the example above). Determining the status of a norm thus requires examining the interactions between multiple norms, and given a system containing many norms, it can be difficult for a user or designer to identify why some norm is assigned a certain state. A graphical model for norms would allow for such links to be made explicit. More generally, humans are able to assimilate large amounts of graphical information, and thus, by modelling norms graphically, the norm system can be more easily understood. Our chosen graphical formalism is based on conceptual graphs, due to their well understood nature and

formal semantics. Having provided an overview of our normative model, we now proceed to describe conceptual graphs in more detail.

2.2 Conceptual Graphs

Due to their visual qualities, semantic networks, which were originally developed as cognitive models, have been used for knowledge representation since the early days of artificial intelligence, especially in natural language processing. They all share the basic idea of representing domain knowledge using a graph, but there are differences concerning notation, as well as representational and reasoning power supported by the language. In semantic networks, *diagrammatical reasoning* is mainly based on path construction in the network. We can distinguish two major families of logically founded languages born from semantic networks: KL-ONE and conceptual graphs. KL-ONE [12] is considered to be the ancestor of description logics (DLs) ([1]), which form the most prominent family of knowledge representation languages dedicated to reasoning on ontologies. However, DLs have lost their graphical origins. In contrast, conceptual graphs (CGs) were introduced by Sowa (cf. [10, 11]) as a diagrammatic system of logic with the purpose "to express meaning in a form that is logically precise, humanly readable, and computationally tractable" (cf. [11]). Throughout the remainder of this paper we use the term *conceptual graphs* to denote the family of formalisms rooted in Sowa's work and then enriched and further developed with a graph-based approach (cf. [5]).

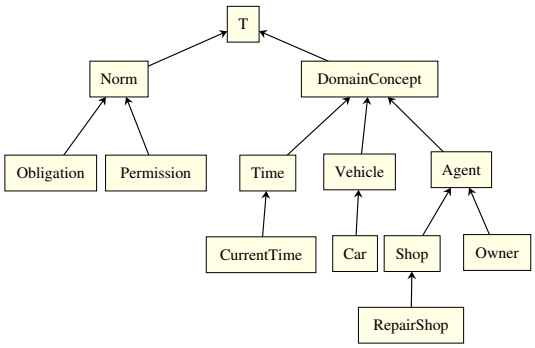


Fig. 1 Conceptual graph support: the concept hierarchy.

Within the conceptual graph approach, all types of knowledge are encoded as graphs and can thus be visualised in a natural way. A CG partitions knowledge into two types. The first type identifies the CG's *vocabulary* and can be seen as a basic ontology. This vocabulary is composed of hierarchies of concepts and relations, which are referred to as the CG's *support*. Since both the concept support and relation support are partial orders, they can be visualised by their Hasse diagram. The

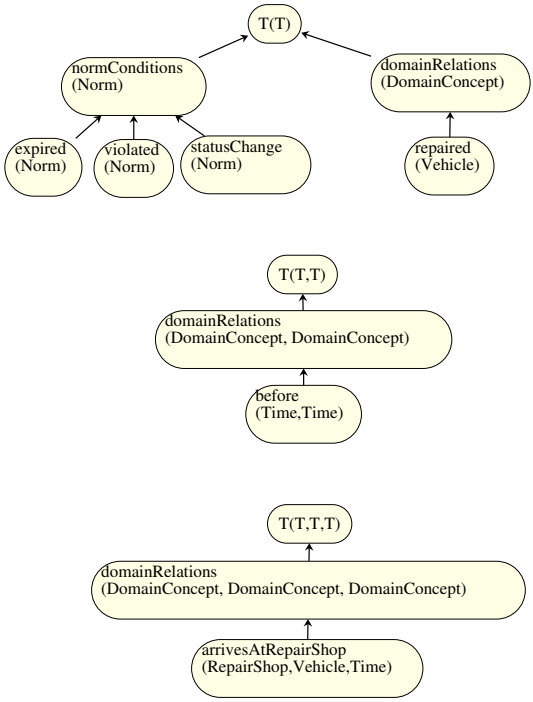


Fig. 2 Conceptual graph support: the relation hierarchy.

partial order represents a specialisation relation. If $t' \leq t$ within the partial order, then t' should be interpreted as a specialisation of t. Figures 1 and 2 respectively illustrate the concept and relation hierarchies that are used to represent norms and domain concepts in the repair shop example throughout this paper.

The second type of knowledge encoded within a conceptual graph is based on the representation of entities and their relationships, encoded by a labelled graph with two kinds of nodes (corresponding to entities and relations). Edges link an entity node to a relation node, and such nodes are labelled by types of the vocabulary. Concept nodes are normally drawn as rectangles and relation nodes as ovals, while the edges incidental to a k-ary relation node are numbered from 1 to k. Figure 3 presents an example of this type of graph, which encodes the fact that a car arrived at the *repairShop* at some time. This second type of graph is called a *basic graph* (abbreviated BG) in the CG literature.

Having described the (graphical) syntax of a CG, we now proceed to look at its semantics. These semantics are in fact defined using first order logic, as defined by a mapping classically denoted by Φ in the conceptual graphs literature [11].

More specifically, let G and H be two BGs. A *homomorphism* π from G to H is a mapping from the concept node set of G to the concept node set of H and from the relation node set of G to the relation node set of H, which preserves edges and may decrease concept and relation labels, that is: *(i)* for any edge labelled i between the

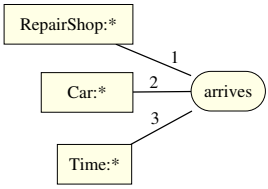


Fig. 3 A generic basic conceptual graph fact.

nodes c and r in G, there is an edge labelled i between the nodes $\pi(c)$ and $\pi(r)$ in H; *(ii)* for any (concept or relation) node x in G, the label of its image $\pi(x)$ in H is less than or equal to the label of x.

The fundamental theorem states that given two BGs G and H, **there is a homomorphism** from G to H if and only if $\Phi(G)$ is a **semantic consequence** of $\Phi(H)$ and the logical translation of the vocabulary, i.e. $\Phi(\mathscr{V}), \Phi(H) \models \Phi(G)$ (i.e., this is a soundness and completeness theorem of BG homomorphism with respect to first order logic entailment). It should be noted that BGs are in fact equivalent to the positive, conjunctive and existential fragment of first order logic.

Having described our the normative model and introduced conceptual graphs, we now proceed to detail how a norm can be represented within a CG based framework.

3 Graphically Computing the Status of Norms

3.1 Modelling Norms with CGs

We will represent both abstract and instantiated norms using a tree structure, referred to as the *norm tree*. The root of the node tree represents the entire norm (by capturing its type and target), while lower levels of the tree represent different portions of the norm. Nodes in the second level of the norm tree are associated with the activation condition, while nodes in the third level are associated with the normative condition, and in the fourth level with the expiration condition. Figure 4 depicts a norm tree. Each of the nodes in the norm tree has associated a conceptual graph representation of their content.

The nodes of the norm tree are used to represent disjunctive conditions of the appropriate portion of the norm. More formally, we assume that the norm target parameter consists of a conjunctive combination of predicates, and that all other parameters (except for norm type) may contain disjunctions. In order to map a norm into a norm tree, we represent the norm using the disjunctive normal form of its elements, i.e. a norm tuple $\langle Type, AC, NC, EC, NT \rangle$ can be rewritten as

$$\langle Type, \bigvee_{i=1,a} AC_i, \bigvee_{j=1,c} NC_j, \bigvee_{k=1,e} EC_k, NT \rangle$$

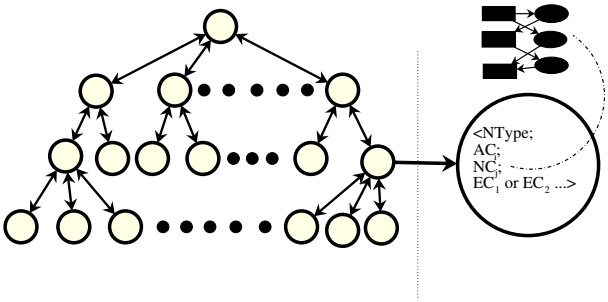


Fig. 4 A conceptual representation of a norm tree

Where $Type$, AC_i,NC_j, EC_k and NT are all conjunctive positive existential first order logic formulae. It should be noted that this requires the assumption of negation as failure. We may then represent each of these formulae as a conceptual graph, defined on some given support (i.e. domain ontology).

Then a norm tree T_{n_1} for norm n_1 is defined as follows:

1. The root of the tree is a node containing n_1. The node is labelled with $Type$ and NT.
2. Each child of the root (i.e. each node at level one) contains a norm n_1^i for node $i = 1 \ldots a$ of the form

$$\langle Type, AC_i, \bigvee_{j=1,c} NC_j, \bigvee_{k=1,e} EC_k, NT \rangle$$

   The node containing n_1^i is labelled with AC_i.
3. Each node at level two which is a child of n_1^i contains a norm n_1^{ij} for $j = 1 \ldots c$ of the form

$$\langle Type, AC_i, NC_j, \bigvee_{k=1,e} EC_k, NT \rangle$$

   The node containing n_1^{ij} is labelled with NC_j.
4. Each node at level three which is a child of n_1^{ij} contains a norm n_1^{ijk} for $k = 1 \ldots e$ of the form

$$\langle Type, AC_i, NC_j, EC_k, NT \rangle$$

   The node containing n_1^{ijk} is labelled with EC_k.

Each node in the tree is associated with a conceptual graph.

Let us consider the norm presented in Section 2.1 stating that a repair shop has an obligation imposed upon it, to repair a car within seven days of its arrival. Figure 5 illustrates the simplified norm tree[2] that is associated with this norm. The top node

[2] In the remainder of this paper, and unless otherwise stated, we ignore the norm target parameter, assuming it is present in the root node.

represents the type of the norm. The second level of the tree depicts the norm's activation condition, while the third level represents the normative condition.

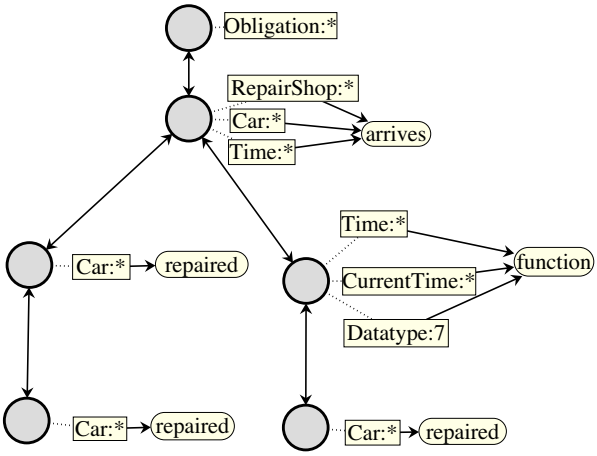


Fig. 5 The Norm Tree for the repairshop example norm.

There is a separation between the semantics of the normative model and its norms, and those of the knowledge base system. For a parameter (such as the normative condition) in the norm to evaluate to true, any of the disjunctions from which it is composed must evaluate to true (e.g. *repaired(Car)* in the above example). This aspect of a norm is captured by the normative model's semantics, and is thus represented by the norm tree structure. However, reasoning within the knowledge base system is kept separate from these norm model semantics by means of conceptual graph annotations of the nodes in the normative tree. Thus, the knowledge base system will identify *which* of the normative conditions disjunctions actually evaluated to true in the case where the normative condition is true. This is done by means of colouring; black nodes indicate unsatisfied conditions, while grey nodes indicate satisfied normative conditions. If at least one node is satisfied at some level of the norm tree, that condition is deemed to have been satisfied (thus, for example, if at least one node at the third level of the tree is not black, the normative condition for the norm is met).

Finally the last level of the tree depicts the expiration condition. It is assumed that the norm target attribute is used to retrieve the literal corresponding to the agent name upon whom the norm is imposed, and we thus assume it forms part of the tree's root node. However, future work will look into retrieving agents by their type (e.g. which obligations are imposed on agents of type *shop*, which may include more specific agents of type *repairShop*, *groceryStore* and so on). In this case the graph based representation of the support will also provide useful feedback to the user (for example, identifying that an agent was selected as its type is a descendant of the *repairShop* node in the ontology).

The conceptual graph representation provides us with two advantages over a textual representation of the norm. First, the conceptual graph representation makes visually explicit the types of the concepts linked up by predicates (*RepairShop*:* as opposed to *X*). While this problem is easily addressed by manually changing the variable names of the textual logic representation (using "meaningful" literals), the heuristic employed could be confusing (e.g. a variable label such as *RepairShopNumber* could imply a certain ordering of the variables etc.). Second, and more importantly, for elaborated pieces of knowledge (namely conjunctions with common variables) the translation between natural language and logical formulae becomes very difficult. For example let us assume we are trying to represent the fact that a car arrives at a repair shop that accepts only Volvos, and the time when the car arrives at the repair shop has to be later than 9 (the shop's opening time). This norm requires reasoning about different ontological levels as well as the logical formulae representing the norms, and a textual logic based representation of this norm can be difficult to follow. The conceptual graph depiction of this type of norm is visual, and thus more intuitive.

3.2 Instantiating Norms

Figure 5 represents an abstract norm. Now assume that a number of new facts are added to the knowledge base, namely that some car, c_1 arrived at Bob's repair shop at time 12. In predicate form, we write $arrivesAtRepairShop(bob, c_1, 12)$.

This piece of knowledge will be projected to all the norm conditions in the system. The mapping will instantiate a number of generic nodes in the conceptual graphs annotating the norm tree nodes. In this way we obtain the instantiated norm shown in Figure 6. For clarity, we differentiate between norm trees for instantiated and abstract norms by colouring the nodes of the latter in white, and of the former in grey or black (depending on whether they are satisfied or not).

3.3 Computing the Status of Norms

So far, we have shown how abstract and instantiated norms may be represented as norm trees. The main focus of the framework presented in [8] revolved around norm status monitoring (i.e. identifying when a norm has a specific status such as complied with or expired), and we now discuss how a norm's status may be identified using the norm tree structure.

As new facts appear and disappear within the knowledge base, the status of norms will change. Computing these statuses is done by checking for the existence of projections between the facts in the environment and the conceptual graph annotations of the norm tree. The norm tree on the left of Figure 7 contains a mixture of black and white nodes. The white node corresponds to the fact that the node is satisfied,

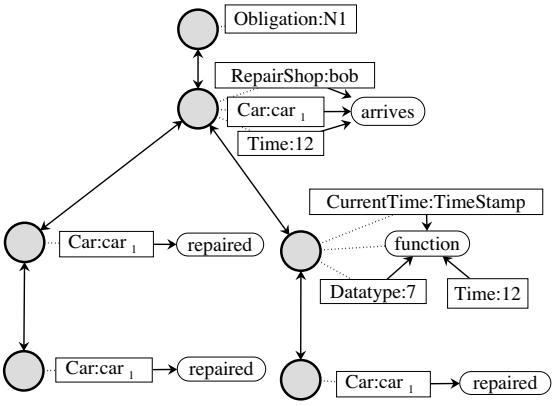


Fig. 6 An instantiated norm for the repairshop example.

e.g. there is a projection between the environment (on the right hand side) and the corresponding CG annotation. The other nodes are black: they are not satisfied. Thus, for example, there is no projection between the CG node representing the expiration condition, which states that the car is repaired, and the CG on the right of Figure 7. Similarly, there is a projection (and thus the node is white) between the CG on the right, and the CG captured by the node at the normative condition level stating that the current time is before 19 (the condition in this latter node is represented by the function taking in the datatype, time and current time). If, at some later point, the car is repaired, the black nodes within the norm tree will turn white.

During its lifecycle, an abstract norm becomes instantiated. While instantiated, its normative condition may evaluate to true or false at different times. Finally, the norm's expiration condition evaluates to true, after which the instantiated norm is deleted. We have already seen how one may determine whether a norm may be instantiated using a norm tree. A norm's normative condition is met, i.e. evaluates to true, if any of the nodes at the norm condition level are white. Similarly, a norm expires if any of the nodes at the expiration condition level are white.

A norm's status includes whether it is activated or expiring, and whether it is being met, and it is easy to determine this from the norm tree. It is also possible to identify more complex norm statuses. For example, a norm is said to be *violated* if it is an obligation which has been instantiated, and whose normative condition evaluates to false. It is possible to construct this condition as a query to the knowledge base, and from this, visually determine whether the norm is violated or not.

4 Discussion

Much of the existing work on norms and normative reasoning originated from the philosophical domain. Such work, while recognising the conditional nature of

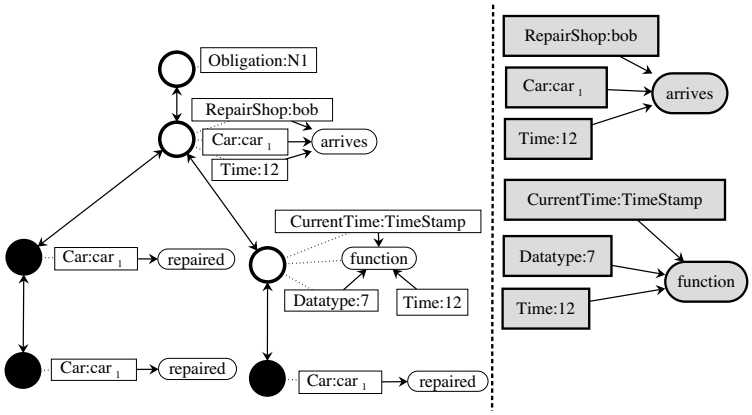


Fig. 7 A norm tree whose nodes are evaluated according to the knowledge base shown on the right.

norms, emphasised problems such as identifying what state of affairs should hold, or how to resolve normative conflict. However, apart from the work of Governatori [6], few have considered how a normative system evolves when norms are fulfilled. Governatori adopts a defeasible logic based approach to norm representation, with norms expiring when a defeater to them is introduced. Within a long lived system, this approach is cumbersome; reinstantiating a norm requires the introduction of a defeater to the defeater. The framework presented in this paper is intended to capture the evolution of a norm over time, allowing for its instantiation and expiration, as well as recording the time periods during which a norm was complied with or violated. Since the internal structure of such a norm is somewhat complex, some technique for explaining why a norm is in a certain state is required, and we proposed a visual model for explaining the status of a norm. The ability to provide explanations for a norm's status in such domains is particularly useful. For example, complex contract disputes may require that some rewards or penalties be assigned by a human mediator, but in order to perform this assignment, the mediator must first understand which norms were violated, and which were complied with. Norm explanation is also important at the system design stage, where an understanding of norm statuses in different situations is needed to ensure correct system behaviour.

We are aware of very little work dealing with the explanation of norms to users. This may be due to an implicit assumption that normative systems are fully automated, and that explanation is thus not necessary, or perhaps due a presumption regarding the technical expertise of the system's users. However, even if a user is able to understand a norm representation, when reasoning about complex interactions between large groups of norms, graphical explanations may be advantages. The work described in [7] touches on the concept of norm explanation. Here, norm violation is analysed and explained by means of a causal graph. The causal graph was then further processed to identify whether mitigating circumstances existed for the norm's violation, and norm explanation was thus not the focus of that work.

In this paper we described how a rich model for tracking and determining the status norms may be represented graphically. As a norm's status changes, so does its graphical representation. This allows the normative system to be understood visually. The use of conceptual graphs to provide the formal underpinnings of our representation will allow us to extend this work in a number of interesting directions. While other studies have shown that graphical representations are more easily understood by non-experts than logic based ones [5], we have not yet evaluated our model in this way, and intend to do so in the short term. We also intend to leverage the formal power of our model, by investigating the use of graph theoretical operations to identify redundant norms [2]. Similarly, we believe that graph based operations can be used to detect, and help resolve, normative conflict. Finally, we intend to investigate more complex norm statuses than the ones described in this paper. For example, a more complete model of obligation violation requires determining whether a permission, acting as an exception to the obligation, exists. Here, complex interactions between more than one norm must be considered, and graphical models are ideal for reasoning about, and explaining such interactions.

Acknowledgements The authors would like to thank the EU Agreement Technologies COST action for providing a STSM grant which made this work possible.

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. G. Boella and L. van der Torre. Permissions and obligations in hierarchical normative systems. In *Proc. of ICAIL 03*, Edinburgh, Scotland, 2003.
3. G. Boella and L. van der Torre. Institutions with a hierarchy of authorities in distributed dynamic environments. *Artificial Intelligence Law*, 16:53–71, 2008.
4. W. Briggs and D. Cook. Flexible social laws. In C. Mellish, editor, *Proc. of the Fourteenth Int. Joint Conf. on Artificial Intelligence*, pages 688–693, San Francisco, 1995. Morgan Kaufmann.
5. M. Chein and M. Mugnier. *Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs*. Springer, 2009.
6. G. Governatori, J. Hulstijn, R. Riveret, and A. Rotolo. Characterising deadlines in temporal modal defeasible logic. In *Proc. of AI-2007*, volume 4830 of *Lecture Notes in Artificial Intelligence*, pages 486–496, 2007.
7. S. Miles, P. Groth, and M. Luck. Handling mitigating circumstances for electronic contracts. In *AISB 2008 Symp. on Behaviour Regulation in Multi-agent Systems*, pages 37–42, 2008.
8. N. Oren, S. Panagiotidi, J. Vazquez-Salceda, S. Modgil, M. Luck, and S. Miles. Towards a formalisation of electronic contracting environments. In *Proc. of Coordination, Organization, Institutions and Norms in Agent Systems, the International Workshop at AAAI 2008*, pages 61–68, Chicago, Illinois, USA, 2008.
9. Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, 73(1–2):231–252, 1995.
10. J. F. Sowa. Conceptual Graphs. *IBM Journal of Research and Development*, 20(4):336–375, 1976.
11. J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.
12. W. Woods and J. Schmolze. The kl-one family. *Computers Math. Applic.*, 23:133–177, 1992.

Modelling Social Structures and Hierarchies in Language Evolution

Martin Bachwerk and Carl Vogel

Abstract Language evolution might have preferred certain prior social configurations over others. Experiments conducted with models of different social structures (varying subgroup interactions and the presence of a dominant interlocutor) suggest that having isolated agent groups rather than an interconnected agent is more advantageous for the emergence of a social communication system. Accordingly, distinctive groups that are closely connected by communication yield systems less like natural language than fully isolated groups inhabiting the same world, while the addition of a dominant male who is asymmetrically favoured as a hearer, and equally likely to be a speaker has no positive influence on the quality of the emergent communal language.

1 Introduction

The question of how human language could have emerged from an animal-like communication system is not only fascinating from an evolutionary point of view, but also has broad ramifications in the area of natural language and speech development. If we could understand how our extremely distant ancestors learned to associate meaning with seemingly arbitrary symbols, be those symbols gestures or sounds, then we should have an easier time of engineering artificial systems capable of comparable levels of intelligence.

Although speculation about the origin of human language has gone on for centuries, the problem has only relatively recently been scrutinized in empirically ori-

Martin Bachwerk
Computational Linguistics Group, School of Computer Science and Statistics
Trinity College, Dublin 2, Ireland, e-mail: bachwerm@cs.tcd.ie

Carl Vogel
Computational Linguistics Group, School of Computer Science and Statistics
Trinity College, Dublin 2, Ireland, e-mail: vogel@cs.tcd.ie

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_4,

ented disciplines, including anthropology and evolutionary biology [7], linguistics [12], artificial intelligence [20, 6], and computer science [14, 19]. The contribution of the latter two sciences to the problem has been mostly in the area of modelling and simulations, concentrating on collecting experimental data for the plausibility of some of the proposed theories of language evolution. As helpful as such work might be, it has to be noted with some disappointment that few of the computational approaches mentioned above have ventured deeper into isolating the importance of the still obscure issues of symbol grounding, dialogue structuring, questions, mental representations and pragmatics with a realistic set of assumptions.

The current work aims to contribute in this direction with model that is quite abstract, yet sufficiently realistic in terms of assumptions made regarding the cognitive capacities of early humans. We apply this model to a number of different hypothetical social structures among groups of agents in order to observe the communicative advantages and disadvantages of these structures with respect to supporting the emergence of natural language. First we summarize our modelling approach and then provide a technical description of the class of models explored here. We present some of our most recent experiments and discuss expected and observed outcomes, concluding with an evaluation of the results and suggestions for future work. While the results are not decisive, we hope to intrigue the reader with the overall approach and motivate our colleagues to adapt the approach to other related scenarios.

2 Modelling Approach

The experiments presented in this paper have been performed using the Language Evolution Workbench (LEW, [23]), which was extended by a more intuitive forgetting mechanism as well as the possibility to run simulations with different underlying social structures, as explained below. This workbench provides over 20 adjustable parameters and makes as few *a priori* assumptions as possible. The nevertheless assumed cognitive and social abilities of agents have been motivated by the more widely accepted evolutionary accounts and thus fit in well with a number of models and scenarios proposed by other authors (see Section 1 for a small selection of literature).

Abstract Model The LEW model is implemented at a relatively high level of abstraction, meaning that some interactions and factors of the outside world are either modelled in a very simplistic manner or not modelled at all. While we concede that such an approach might make the model open to a certain amount of criticism regarding its validity in terms of being an acceptable representation of reality, there are two arguments that should be mentioned in defence of such an approach.

First of all, a highly abstracted model of a certain system means that all the elements of such a model are distinctly observable and their effects well quantifiable. While a model with hundreds of parameters would certainly bring it closer to reality, one would find it extremely hard to distinguish between significant and insignificant

parameters in such a model, as well as to observe the interactions between different parameters.

Furthermore, by starting with a simpler model, we aim to avoid the mistake of building features into it that have not yet been proven or observed well enough in other disciplines. In other terms, if one does not know the precise parameter settings for the dimensions that impinge on the problem, one should not just build in arbitrary settings as features of the model without experimenting with a range of parameter combinations first. However, since the number of such experiments grows exponentially with every free parameter in a model, we have elected to approach this issue by tackling a smaller number of features at a time, with the option of fixing the parameter values of a particular feature in case of little or no significance and moving on to the next feature, thus gradually extending the model.

Due to the abstracted nature of agent, entity and event representation, it should be noted at this point that the model is easily adjustable to represent a wide variety of social scenarios, thus making it well suited for experiments even outside the scope of language evolution. The main emphasis of the model is on observing how patterns emerge in a simulated system without there being any sort of explicit force driving the system in any particular direction.

General Assumptions Agents in the LEW are equipped with the ability to observe and individuate events, i.e. an abstracted sensory mechanism. Each agent individuates events according to its own perspective, as likely as not distinct from that of companions. In order to model communication, agents are assumed to be able to join in a shared attention frame around an occurring event and engage in an interaction, whereby one of the agents is assigned the intention to comment on the event and the other, listening agent understands that the observed utterance is the speaker's comment on the event and attempts to decode the meaning of the perceived symbols accordingly. These cognitive skills of attention sharing and intentionality perception have been marked as integral to the origins of language among others by [21].

Three further assumptions are relevant to symbol production and perception during interactions between agents: that agents are able to produce discernible symbols at all, that such phonemes can be combined to invent new symbols and that the transmission of symbols and phonemes occurs without noise; however, agents do not necessarily segment symbol sequences identically.[1] These assumptions are made on the grounds that language could not have possibly evolved without some sort of symbols being emitted.

The LEW fits with the so called faculty of language in the narrow sense as proposed by [11] in that the agents are equipped with the sensory, intentional and concept-mapping skills at the start, and the simulations attempt to provide an insight into how these could be combined to produce a communication system with comparable properties to a human language. Further, the LEW agents can be seen as having

[1] While the symbols are called phonemes in the current simulations, there is no reason why these should not be representative of gestural signs. However, the physiological constraints on the production of symbols is not a part of this model.

completed steps 1 and 2 in the accounts presented by [13] or [4], i.e. autonomously re-using and inventing new symbols from a generative unit, the phoneme.

Social Structures and Hierarchies The notion that social groups of one type or another play a central role within the evolution of the hominid species as such as well as the emergence of a communicative system like a proto-language in particular is apparent from a variety of evolutionary theories and modelling approaches. From the anthropological point of view, it has been repeatedly suggested that the emergence of language is strongly connected with the increase of hominid group sizes and the directly related neocortex ratio between 500,000 and 250,000 years ago (cf. [1]). Being unparalleled in any other species, this evolutionary change has become the focal point of several theories on the emergence of language. While the specific details of these theories are quite variable, two main branches can be clearly distinguished in terms of the characteristic social dynamics of the scenarios.

Nearly 40 years ago, [2] postulated that the unparalleled evolutionary path of hominids is based mainly on the competition between different bands or groups of the species. While this phenomenon has been also observed in other primates to some extent, the degree of competitiveness, escalating to true warfare, is considered to be unique to the human species. In contrast to [2], [7, 6] propose scenarios that are based on the evolution of *Machiavellian Intelligence* in early hominids [3]. The main difference to [2] is that the focal point in these scenarios is on group-internal organisation and cooperation, rather than inter-group competition. Accordingly, internal hierarchies play a much bigger role in these accounts, even if considered at the simplest level of having one dominant member in a group. In the current experiments, the groups in the no-male runs should roughly correspond to the competing bands in [2] and the simulations with a male – to the social structures in [7, 6].

One final remark regarding our implementation of 'competition' or an 'alpha-male' is that even though it is common in social and political sciences to observe a distribution of power and influence in basically any community, our model does not involve an explicit definition of power. Consequently, we can observe the effect of being organised in a 'democratic' or an 'dictatorial' power structure, as proposed by [8], only in approximated terms of implicit influence, i.e. based on some agents' higher involvement in interactions, meaning there is never a 'semantic arbiter'.

3 Model Implementation

Agents, Entities and Events Agents in the LEW are non-physical entities (cf. [20] for embodied implementations) and are not specialized to the question of language evolution. What characterizes every agent in the LEW is solely a knowledge base and a lexicon. The knowledge base consists of all experienced events in the order in which an agent encountered these. The lexicon is represented as a set of $<$*Meaning*, *Form*, *Weight*$>$ tuples, where a *Meaning* is (a part of) an event, a *Form* is (a part of) an utterance that was either produced or heard by the agent in relation to the event,

and the *Weight* is an indicator of confidence in the mapping, incremented each time it is experienced. If forgetting is enabled, these weights are then gradually decreased according to the selected forgetting function and its parameters.

Events are generated by selecting one of the predefined *event types*, which define the combination of arguments that is permitted for event instances of the given type (e.g. `[human, human, event]`), and filling it with acceptable arguments. An *argument* of an event can either be an entity or another event, the latter option allowing for recursive composition of events, resulting in an unbounded meaning space based on a finite number of event types. Entities are represented as property-less atoms, and an arbitrary number of these can be experimented with. However, in the presented work we define entities in terms of sorts, whereby two sorts are distinguished, namely `animates` and `inanimates`. Adding an abstracted layer of physical properties to entities for simulating concept formation is a possible future extension of the LEW.

Interactions Building on the traditions of computer simulations of language evolution, the LEW simulates interactions between agents. Every interaction in the LEW occurs between two randomly chosen agents, a speaker and a hearer, whereby an agent can also end up talking to himself if he gets picked as the hearer too (language is meant for thinking as well as communicating). The speaker is first of all presented with an event constructed as described above, e.g. `xcvww human twedf inanimate`. The speaker's task is then to individuate the event by segmenting it into meaning chunks or, in other terms, by combining parts of the event into unified segments, e.g. `[xcvww] [human] [twedf inanimate]`, which he then attempts to communicate to his conversation partner by either using an appropriate mapping from his lexicon or by inventing a new word if the meaning is new to him.

The second agent – the hearer – has the task of decoding the meaning of the heard utterance by attempting to assign (parts of) the event to (parts of) the utterance by either looking for appropriate form-meaning pairs in its lexicon or, failing to find one, by simply assuming (a part of) its own perspective on the event (e.g. `[xcvww human twedf] [inanimate]`) as the intended meaning. While this scenario presumes that both agents are knowingly communicating about the same event, their internal segmentations of the event can be, and usually are, quite different, which ensures that no omniscient meaning-form transfer occurs at any stage of the simulated interactions.

The words used by the agents in their interactions are implemented as combinations of phonemes, whereby every phoneme is represented as a pair of phones, thus mimicking the onset-nucleus structure (without the coda). When inventing a new word, the speakers use a single phoneme only. However, since agents do not possess the capacity of detecting word boundaries from an encountered utterance, hearers have the 'power' to wrongfully segment heard utterances and thus introduce

larger words into their lexicons and subsequently, when acting as a speaker, into the lexicons of others.[2]

Group Dynamics In order to be able to perform experiments with different social structures as described in Section 2, we have extended the LEW with three parameters that determine the social organisation of a simulated population, namely the presence of a 'male' (represented as a binary variable), the number of groups n (≥ 1) that the non-male population should be split up into and the ratios for the distribution of agents into these groups $r_1, .., r_n$, so that the size of any non-male group $C_i = r_i * (C_{total} - male)$, where C_{total} is the total number of agents in the system.

After dividing the agents into a particular social structure, we can define how they will interact with each other during the simulation using two further parameters: the male-directed communication rate p_{male}, defines the chances of an agent selecting the male as the hearer in an interaction, and the intra-group communication rate p_{intra}, which is defined as the probability of a speaker agent from group G_i picking another agent from his own group (including himself) as the hearer, as opposed to an agent from groups $G_1..G_{i-1}, G_{i+1}..G_n$, after having decided that he does not wish to interact with the male. The probabilities of picking an agent from either of the neighbouring groups were distributed equally from any remaining percentage.

Table 1 Probabilities of speaker-hearer combinations for each type of agent.

Speaker \ Hearer	Male	Same group	Every other group
Male	0	–	$\frac{1}{n}$
Non-Male	p_{male}	p_{intra}	$\frac{1-(p_{male}+p_{intra})}{n-1}$

Importantly, and as can be seen in Table 1, the intra-group communication rate only applies to non-male agents, meaning that the male has equal chances of selecting any agent from any group for an interaction, except for himself. The avoidance of male self-talk is mainly motivated by the fact that a male is already involved in a much larger number of communicative bouts and may simply not have enough time to be alone and talk to himself.

4 Experiment Design

The goal of the presented experiments was to observe the effect of different hierarchies and social structures on the overall speed and success of communication within a group of agents. This approach extends the LEW in a way that would enable it to be used at least as a partial model for the theories of the origins of language that are based on social interactions of early humans. In particular, the experiments

[2] The possibility of having synchronized speech segmentation can nevertheless be explored in the LEW via the synchrony parameter, but was turned off for the presented experiments (cf. [22] for an account of experiments with synchronous transmission).

should provide empirical data for the possibility of language emerging in differently organized social groups, building on either the competitive account presented by [2] or the grooming theory of [7] and the corresponding comparative research by [15].

For the current experiments, all but two parameters of the LEW have been kept fixed at the following values: 9 agents divided equally into three groups of 3, with no agent addition or elimination occurring, the male-directed communication rate set to 20% in all simulations where a male was present, 100 event types with a Zipfian distribution, 41 phonemes, asynchronous utterance segmentation, frequency-based mapping retrieval, forgetting enabled and no questions. The two varied parameters were the presence of a male in a population and the intra-group communication rate for the non-male agents whereby five different rates were experimented with – 0%, 33%, 50%, 80% and 100% – resulting in ten combinations of simulation settings that represent a variety of social structures from free circulation to full isolation.

Expected Outcome The goal of the experimental setup described above was to observe if a particular social structure is somehow better suited for the emergence of a group-wide communication system. The prediction that we make is that agents communicating mainly within their own group should achieve higher levels of understanding; however, these agents are expected to evolve their own sub-dialects that are quite distinct from those of other groups of the community, thus making them unable to properly cooperate with most members of the whole community, if the need for such cooperation ever occurred. The evaluation measures applied during the experiments in order to verify the postulated hypothesis and quantify the suggested effects are described in more detail below.

Evaluation Measures When two agents communicate with each other in the LEW, they have no access to the internal states of their interlocuting counterparts and are thus, rather disappointingly from their point of view, unable to telepathically know what the other agent is talking about. However, this does not mean that communication success is not measurable in some way. In fact, the model includes a number of various measures that allow us to observe and analyse the emerging symbol systems in a sufficiently rigorous fashion, for example by comparing the intended meaning of the speaker with the understood meaning of the hearer, either explicitly or implicitly.

From the explicit point of view, one can observe how many of the speaker's words have been actually segmented correctly, and subsequently how many of the correctly segmented words have been decoded into a meaning by the hearer that matches exactly the intended meaning of the speaker. This explicit measure can be also seen as measuring the cohesiveness of lexical overlap in two interacting agents. However, understanding can be also measured implicitly, namely without regard for the lexical items that were used to convey the meaning. So if the speaker wants to say *A*, but either by accident or because of not knowing any better, says *B*, then if the hearer, again by chance or lack of linguistic knowledge, still understands *A* then the interaction can be seen as successful to a certain degree.

Apart from evaluating the actual communication scenarios, we also observe the lexicons of the agents to be able to draw more qualitative conclusions. For instance,

the lexicon size indicates the range of expressible meanings and interpretable forms; the amount of synonymy and homonymy both inside the individual lexicons and across of the whole population tells us how similar the emergent languages are to natural languages, which seem to tolerate homonymy and avoid synonymy; while the amount of mappings shared by the whole population and the number of agents sharing a mapping on average are both good indicators of potential communicative success.

5 Results and Discussion

In total, 600 runs have been executed for each of the factor combinations with a total of 200 rounds of 10 interactions within each such run. The distributions of understanding success rates are presented in Figure 1(a) and suggest that there is a strong difference in the potential of language evolving in a particular group depending on the group's hierarchical and social structure.[3] In particular, one observes an increase in communicative success with the increase of intra-group communication rate ($t \geq 14.10$, $p < 0.0001$ for every level of p_{intra}), showing that the more isolated a group of agents is, the likelier it seems to evolve a group-internal language that serves well as a medium of communication. It should be noted that the setting with no male and a 33% intra-group communication rate corresponds to one group of 9 agents where any agent has an equal chance of being selected as either the speaker or the hearer, and is thus essentially the current baseline of the LEW as all previously conducted experiments were performed with this setup [23, 22].

When considering the effect of having a male in the population, it can be noted that the addition of such a single interconnected agent is not advantageous for the whole group ($t = -12.43$, $p < 0.0001$). Most plausibly, this can be explained by the lack of opportunity for the agents to build a common language as they are in effect too occupied with attempting to communicate with the male. However, since the chances of the male being selected as the speaker are equal to those of any other agent, he does not have sufficient power to actually regulate and stabilize the language of other agents.

Figure 1(b) suggests that the male lexicon in the given scenario and under the presented experimental settings tends to exhibit a larger number of both expressible meanings and interpretable forms, resulting in a bigger size overall, compared to agents from the surrounding groups ($t = 120.44$, $p < 0.0001$). This tendency is more or less a direct consequence of the setting that, regardless of the intra-group communication rate, every agent from any group interacts with the male 20% of the time. Having different members of the community constantly communicating with

[3] Let the reader not be discouraged by the relatively low levels of understanding as it is our foremost goal to show that at least some sort of recognition of what is being talked about can be achieved at the very early stages of language evolution. In addition, [5] suggests that 'nearly' experiences can be quite stimulating even if the actual rewards are minimal.

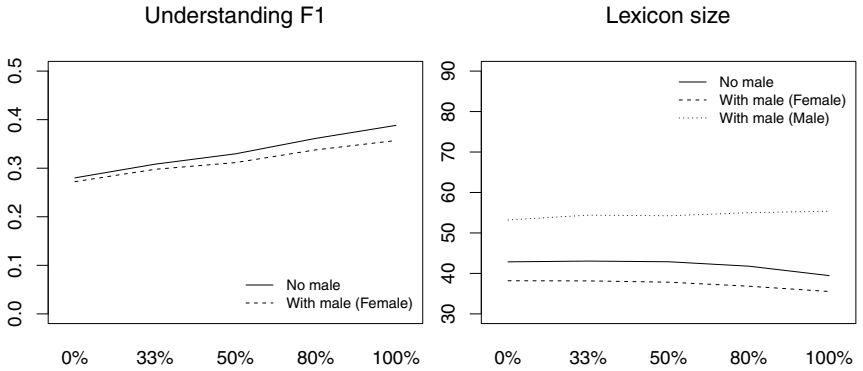


Fig. 1 Effect of male presence and varying intra-group communication rates (x-axis) on (a) communication success rate (F1 measure) and (b) agent lexicon size.

the male results in a deep source of linguistic data being provided to and processed by the male.

A subsequent effect of having a male in the community is that the lexicon size of an average member actually drops below those observed in communities without a unifying male-like agent (t=-33.02, p<0.0001). The explanation is purely quantitative and essentially says that if an agent is occupied with speaking to the male 20% of the time and if the total number of interactions is kept constant, he will have less time to devote to interacting with other members of either his own or neighbouring groups of the community. This in turn reduces the amount of linguistic data that an average group-agent has a chance of observing and learning from, resulting in him learning less meaning-form mappings.

The ramifications of having a male in the presented experiments is both similar and slightly distinct from the acceleration-deceleration effect observed by [10]. On the similar side, the effects of the male acting as a hub and connecting the agents on the one hand and being overloaded with different idiolects and failing to transmit a consistent language to other agents on the other hand are certainly present in our experiments. However, it appears that another complementary effect can be observed in our case, namely that since the chances of the male acting as a speaker are equal to those of any other agent, the male appears to take a lot of linguistic data in, but not give back equally as much, thus acting more like a language server than a true hub.

Going further, a correlation between the higher intra-group communication rates and agent lexicon sizes can be observed in the figures above, most notably for the higher levels of p_{intra} (t=-17.530, p<0.0001 for $p_{intra} = 100\%$ and t=-6.998, p<0.0001 for $p_{intra} = 80\%$; however p=0.337 for $p_{intra} = 50\%$ and p=0.664 for $p_{intra} = 33\%$). It is quite clearly the case that if a regular agent is sufficiently restricted to interacting with other agents of her own group then she is exposed to less linguistic variation and consequently fewer meaning-form mappings that she could possibly learn. On the other hand, the lexicon of the male increases as agents be-

come more and more group-oriented and start developing independent dialects up to the point where they basically have their own languages (t=4.749, p<0.0001 for $p_{intra} = 100\%$) that are only connected by one agent – the male – who is exposed to, and basically forced into learning all three of these. For the fully isolated groups, the smaller lexicons appear to further result in a decrease of agent lexicon synonymy (t=-11.627, p<0.0001); other settings show no such effect.

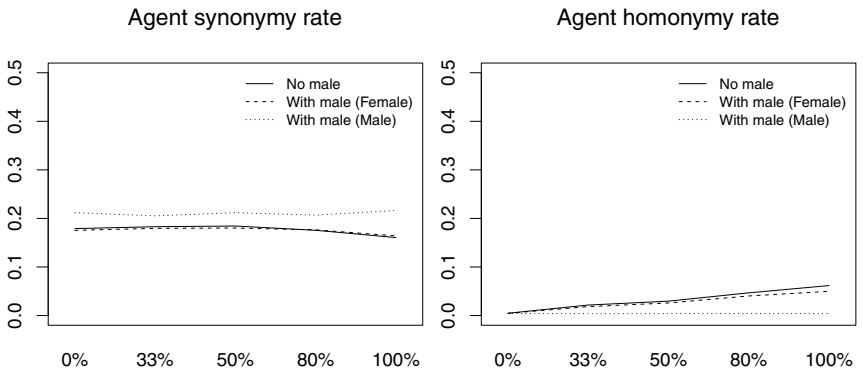


Fig. 2 Effect of male presence and varying intra-group communication rates (x-axis) on (a) agent lexicon synonymy and (b) agent lexicon homonymy.

The explanation for the increase in agent-internal lexicon homonymy corresponding with the increase in the intra-group communication rate requires some clarifications as to how homonyms can emerge in a language. The most intuitive source of homonymy in the LEW is located within the utterance generation process. In particular, when an agent is presented with a meaning that he has not previously encountered and has hence no associated word for, he expresses the meaning by emitting a random string. Importantly, since we are reluctant to assume some kind of a cognitive mechanism implementing the *principle of contrast* as a given, the space of such random strings is not restricted by forms that are already present in an agent's lexicon.[4]

Apart from the above scenario, the current implementation of the LEW also introduces a certain amount of homonymy when an agent engages in a round of self-interaction, which roughly corresponds to the primate notion of 'thinking'. What happens during self-talk is that when an agent assigns some random (even previously unused) form f to a meaning m that he wishes to express, he will not 'remember' that he just assigned that form to the meaning m when acting as a hearer, and if he has never heard the form before, he can end up with selecting any random segment of the perceived event as the meaning, resulting in two different meanings being mapped to a single form within one interaction. This particular 'feature' of

[4] The exact chance of a previously used form being assigned to a new meaning at time t equals the number of forms known by the agent at t divided by the space of possible combinations of two phonemes.

the model explains why the average homonymy levels of agent lexicons tend to increase with higher intra-group communication rates and correspondingly higher self-talk chances ($t \geq 32.78$, $p < 0.0001$ for every level of p_{intra}), as seen from Figure 2(b). The homonymy rates of the male lexicon are resistant to this tendency because the experiments were set up with no male self-talk, motivated in part by them being occupied with speaking to others most of the time.

The initial observations appear to suggest that on an idiolectic level, homonymy seems to creep into an agent's lexicon when the agent frequently engages in rounds of self talk and synonymy is in general governed by the level of isolation of a group from other groups that employ distinct dialects. What can not be concluded from this, however, is if these factors have an equal effect on the synonymy and homonymy ratios of the collective language of the simulated communities. It is not possible to make such conclusions because we can not see the whole picture behind the evolution of meanings and forms from the perspective of any single agent. To exemplify this, it can be imagined that the increase in agent lexicon synonymy does indeed signify an introduction of a number of new and redundant words to the language. However, it can also be the case that the agents within every group previously associated an immensely large amount of words with a few especially frequent meanings, resulting in low lexicon synonymy, but did not know any of the mappings common to the dialects employed by the agents of neighbouring groups, thus making the language of the community possess an extremely high level of overall synonymy. However, if each of these agents were to learn one form for every meaning from each of the other dialects and at the same time forget the redundant synonyms within his own group dialect, global synonymy would actually decrease, courtesy of these forgotten forms, yet the agent lexicon synonymy would increase dramatically.

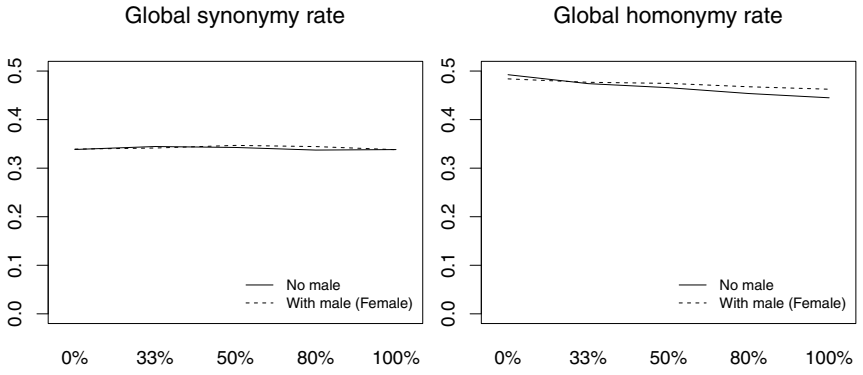


Fig. 3 Effect of male presence and varying intra-group communication rates (x-axis) on (a) global lexicon synonymy and (b) global lexicon homonymy.

This last scenario of increasing idiolectic levels and decreasing global levels does indeed apply to homonymy in the emergent language of the presented simulations

(cf. Figures 2(b) and 3(b)). Accordingly, the global homonymy levels are slightly dropping with the increase of intra-group communication ($t \leq -6.703$, $p < 0.0001$ for every level of p_{intra}), despite the opposite tendency being observed for the individual lexicon homonymy. On the other hand, the effect of group isolation on global synonymy levels appears to be well balanced between lower dialectic and idiolectic synonymy (cf. Figure 2(a)) and the growing idiosyncrasy of the dialects, resulting in an insignificant overall effect ($p > 0.1$ for every level of p_{intra}), as exhibited on Figure 3(a).

6 Conclusions

To summarize the results, three main conclusions can be drawn from the simulation runs described above. The first is that smaller and more isolated groups have a clear advantage over larger or more actively interacting groups in terms of evolving a reliable and useful communication system. The second is that having a single interconnected agent who is asymmetrically favoured as a hearer, and equally likely to be a speaker has no positive influence on the disjoint groups. The third observation is that while the communicative success of agents' interactions in the closely connected groups is inferior to that of the isolated groups: the more 'social' agents appear to learn a significantly larger number of mappings without a commensurate increase in lexical synonymy, resulting in higher chances of understanding utterances by a wider range of agents, including those with different dialects.

Going back to the two general social theories of language evolution presented in Section 2, our results do not truly give support to either of these. In particular, the conclusion that isolated groups tend to outperform more 'social' groups is seemingly at odds with the notion that language evolution was driven by either inter- or intra-group cooperation which lies at the heart of both introduced scenarios. In addition, the theories that are based on group-internal cooperation and should thus involve some sort of hierarchy like having a strongly interconnected alpha-male member are not reinforced by our current simulations either, although we are willing to concede that these structures are still extremely rudimentary and require further investigation.

However, when comparing our results with those of other computational modellers, it appears that the simulations in which agents interact with a higher number of other agents coincide with the results presented by [17, 24], both of which also eschew telepathic meaning transfer and explicit feedback. The experiments conducted by these authors have exhibited similarly high levels of synonymy on both the agent-internal and language-global levels, which is not characteristic of human languages that we know of, leading us to the conclusion that all extant models are still missing some biological or cognitive mechanism that would explain synonymy avoidance as an emergent property, and not a consequence of a pre-programmed principle of discriminative reasoning.

The onus thus remains on the modelling approaches to show that not only can the presented models attest for the emergence of a communicative system, but equally that such a system does indeed significantly resemble language as we know it, which so far is yet to be the case. In particular, models of language evolution that implement such aiding mechanisms like explicit feedback or learning biases like the principle of contrast appear to reliably converge on a perfect communication system without synonyms or homonyms, and without any additional variation after the convergence has been reached, as has been exhibited in [20], as well as numerous related publications. However, while such an emergent linguistic system might seem quite optimal in terms of communicative efficiency, one has to admit that it certainly does not resemble human language.

The shortcomings of such models have been pointed out by several authors, including [22, 17], with [18] attempting to solve the problem by building a model without omniscience or explicit feedback, but with a number of representational, interpretational and social constraints instead, e.g. the "whole object" bias and the principle of contrast. The experimental results of this model suggest that integrating such constraints does indeed improve the communicative success of the emergent language, while keeping it comparable to existing human languages. However, embodiedment of these additional constraints is yet to be conclusively proven to exist in primates or even humans; work by [9, 16] suggests that these constraints are perhaps not present in our brains at all, or at least not in the strong version as implemented in [18].

In conclusion, the experiments presented in this paper hopefully provide an outline of what kind of simulations of group dynamics are possible with the help of the LEW, along with its promise and shortcomings. While the current results are not yet comprehensive enough to speak for themselves in their generality or importance, we hope to have made a case for the experimental approach *per se*. We see this as representative of our programme of experiments on the effects of social structures and group dynamics in language evolution. Future work would certainly benefit from a further extension of the LEW to allow for more complex group dynamics settings, including multi-level hierarchies and coalitions between selected groups.

References

1. Aiello, L.C., Dunbar, R.: Neocortex Size, Group Size, and the Evolution of Language. Current Anthropology **34**(2), 184–193 (1993)
2. Bigelow, R.: The Evolution of Cooperation, Aggression, and Self-Control. In: J.K. Cole, D.D. Jensen (eds.) Nebraska Symposium on Motivation, pp. 1–57. University of Nebraska Press (1972)
3. Byrne, R.W., Whiten, A.: Machiavellian Intelligence: Social Expertise and the Evolution of Intellect in Monkeys, Apes, and Humans. Oxford University Press (1988)
4. Carstairs-McCarthy, A.: The Origins of Complex Language: An Inquiry into the Evolutionary Beginnings of Sentences, Syllables, and Truth. Oxford University Press (1999)
5. Chase, H.W., Clark, L.: Gambling severity predicts midbrain response to near-miss outcomes. The Journal of Neuroscience **30**(18), 6180–7 (2010)

6. Dessalles, J.L.: Why We Talk: The Evolutionary Origins of Language. Oxford University Press (2007)
7. Dunbar, R.I.M.: Grooming, Gossip and the Evolution of Language. Harvard University Press (1997)
8. Gärdenfors, P.: The Emergence of Meaning. Linguistics and Philosophy **16**(3), 285 – 309 (1993)
9. Gathercole, V.C.M.: Lexical Constraints in Acquisition: Are They Viable Any Longer? In: C.M. Gruber, D. Higgins, K.S. Olson, T. Wysocki (eds.) Proceedings of the Chicago Linguistics Society, pp. 481–492. Chicago Linguistic Society (1998)
10. Gong, T., Minett, J.W., Wang, W.S.Y.: Exploring social structure effect on language evolution based on a computational model. Connection Science **20**(2), 135–153 (2008)
11. Hauser, M.D., Chomsky, N., Fitch, W.T.: The Faculty of Language: What Is It, Who Has It, and How Did It Evolve? Science (New York, N.Y.) **298**(5598), 1569–79 (2002)
12. Hurford, J.R.: Social transmission favours linguistic generalization, pp. 324–352. Cambridge University Press (2000)
13. Jackendoff, R.: Possible stages in the evolution of the language capacity. Trends in Cognitive Sciences pp. 272–279 (1999)
14. Kirby, S.: Syntax without Natural Selection: How compositionality emerges from vocabulary in a population of learners, pp. 303–323. Cambridge University Press (2000)
15. Kudo, H., Dunbar, R.: Neocortex size and social network size in primates. Animal Behaviour **62**(4), 711–722 (2001)
16. Quay, S.: Bilingual Evidence against the Principle of Contrast (1993)
17. Smith, A.D.M.: Establishing Communication Systems without Explicit Meaning Transmission. Advances in Artificial Life **2159**, 381 – 390 (2001)
18. Smith, A.D.M.: The Inferential Transmission of Language. Adaptive Behavior **13**(4), 311–324 (2005)
19. Smith, K.: Natural Selection and Cultural Selection in the Evolution of Communication. Adaptive Behavior **10**(1), 25–45 (2002)
20. Steels, L.: Synthesising the origins of language and meaning using co-evolution, self-organisation and level formation. Edinburgh University Press (1996)
21. Tomasello, M.: Constructing a Language: A Usage-Based Theory of Language Acquisition. Harvard University Press (2003)
22. Vogel, C.: Group Cohesion, Cooperation and Synchrony in a Social Model of Language Evolution, pp. 16–32. Springer Berlin Heidelberg (2010)
23. Vogel, C., Woods, J.: A Platform for Simulating Language Evolution. In: M. Bramer, F. Coenen, A. Tuson (eds.) Research and Development in Intelligent Systems, pp. 360–373. London: Springer (2006)
24. Vogt, P., Coumans, H.: Investigating social interaction strategies for bootstrapping lexicon development. Journal of Artificial Societies and Social Simulation **6**(1) (2003)

KNOWLEDGE DISCOVERY AND DATA MINING

On the Usefulness of Weight-Based Constraints in Frequent Subgraph Mining

Frank Eichinger, Matthias Huber and Klemens Böhm

Abstract Frequent subgraph mining is an important data-mining technique. In this paper we look at weighted graphs, which are ubiquitous in the real world. The analysis of weights in combination with mining for substructures might yield more precise results. In particular, we study frequent subgraph mining in the presence of weight-based constraints and explain how to integrate them into mining algorithms. While such constraints only yield approximate mining results in most cases, we demonstrate that such results are useful nevertheless and explain this effect. To do so, we both assess the completeness of the approximate result sets, and we carry out application-oriented studies with real-world data-analysis problems: software-defect localization and explorative mining in transportation logistics. Our results are that the runtime can improve by a factor of up to 3.5 in defect localization and 7 in explorative mining. At the same time, we obtain an even slightly increased defect-localization precision and obtain good explorative mining results.

1 Introduction

Graph mining has drawn a lot of attention recently. One important technique is *frequent subgraph mining* [19], with applications in chemistry and web mining [4] etc. It is often used as a building block of some higher-level analysis task such as *cluster analysis* or *graph classification* [7]. With the latter, frequent subgraph patterns are mined from a set of classified graphs. A standard classifier is then learned on the subgraph features discovered.

Though frequent subgraph mining is an established technique, relying on the pure graph structure is not always sufficient: Many real-world problems correspond to *weighted graphs*. For instance, think of transportation graphs [6]. In software engineering, edge-weighted call graphs (see Figure 1 for an example) have turned out

Karlsruhe Institute of Technology (KIT), Germany
e-mail: {eichinger,matthias.huber,klemens.boehm}@kit.edu

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_5,

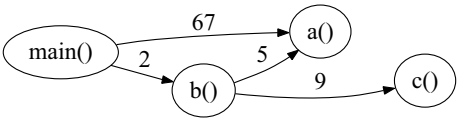


Fig. 1 Example call graph. Edges represent method calls, edge weights call frequencies.

to be beneficial for software-defect localization [1]. To take both the graph structure and the weights into account when mining weighted graphs, one can analyze weights in a preprocessing or in a postprocessing step, before or after the actual (unweighted) subgraph mining takes place [1, 6, 9]. However, both variants have issues: Discretizing numerical values to categorical labels during preprocessing might loose important information, as edges with similar weights can fall into different intervals. Postprocessing in turn is not always efficient. This is because the mining algorithm first ignores the weights and might generate a huge number of subgraphs. The second step however discards most of them.

A cheaper way to perform frequent subgraph mining with weights is *approximate graph mining* [3, 16]. Another approach is *constraint-based mining* [14, 20]. Constraints can be used to prune the search space and speed up mining. We for our part investigate approximate frequent subgraph mining with weight-based constraints. This is promising, since various higher-level analysis tasks imply meaningful weight-based constraints, as we will show. In a classification scenario, to give an example, a natural constraint would demand weights in the subgraph patterns with a high discriminativeness. While constraints lead to smaller result sets, we hypothesize that those application-specific constraints do not lower the result quality of the higher-level problem. However, not every constraint is good for pruning in a straightforward way. Literature has introduced *anti-monotone constraints* [8, 14, 20]: When using them for pruning, the algorithm still finds all patterns. However, most weight-based constraints are not anti-monotone, for the following reason: Graph topology and weights are independent of each other, at least in theory. Example 1 illustrates that weight-based properties of graphs may behave unpredictably when the support changes. Thus, pruning a pattern at a certain point bears the risk of missing elements of the result.

Example 1. Think of an upper-bound constraint defined as a numerical threshold t_u on the average weight of a certain edge $a \rightarrow b$ in all supporting graphs: $avg(a \rightarrow b) \leq t_u$. This would prevent mining from expanding a pattern f where $avg(a \rightarrow b) > t_u$. For the moment, suppose that f was expanded by one edge nevertheless, resulting in pattern f'. Then, fewer graphs in the database might support f'. Depending on the edge weights in the non-supporting graphs, the average weight of that edge in f' might decrease – f' might satisfy the constraint.

Despite this adverse characteristic, we study frequent subgraph mining with non-anti-monotone weight-based constraints in this paper. The rationale is that certain characteristics of real-world graphs give way to the expectation that results are good. Namely, there frequently is a correlation between the graph topology and the weights in real-world weighted graphs.

Example 2. Consider a road-map graph where every edge is attributed with the maximum speed allowed. Large cities, having a high node degree (a topological property), tend to have more highway connections (high edge-weight values) than smaller towns. This is a positive correlation.

In software engineering, a similar observation holds: Think of a node in a weighted call graph representing a small method consisting of a loop. This method tends to invoke a few different methods only (low degree), but with high frequency (high weights). This is a negative correlation.

Our notion of *approximate constraint-based frequent subgraph mining* is as follows: Given a database of *weighted graphs*, find subgraphs satisfying a minimum frequency constraint and user-defined constraints referring to weights. Note that the subgraphs returned are unweighted – weights are considered only in the constraints. In this study, we investigate the following question:

Problem Statement. What is the completeness and the usefulness of results obtained from approximate weight-constraint-based frequent subgraph mining?

In concrete terms, we study the degree of *completeness* of mining results compared to non-constrained results. To assess the *usefulness* of an approximate result, we consider the result quality of higher-level analysis tasks, based on approximate graph-mining results as input.

To deal with this problem, this article features the following points:

Weight-Constraint-Based Mining. We say how to extend standard pattern-growth algorithms for frequent subgraph mining with pruning based on weight-based constraints. We do so for *gSpan* [17] and *CloseGraph* [18].

Application to Real-World Problems. We describe different data-analysis problems that make use of frequent subgraph mining of weighted graphs from domains as diverse as software engineering and logistics. We say how to employ weight-based constraints to solve these problems efficiently.

Evaluation. We report on the outcomes of a broad evaluation from very different domains and analysis settings. A fundamental result is that the correlation of weights with the graph structure indeed exists, and we can exploit it in real-world analysis problems. In particular, graph mining with constraints leads to a speedup of up to 3.5 times with the same quality of results in software-defect localization and 7 times in explorative mining while obtaining satisfactory results.

2 Preliminaries

Definition 1. A *labeled weighted graph* is a six-tuple: $G := (V, E, L, l, W, w)$. V is the set of vertices, $E \subseteq V \times V$ the set of edges, L a set of categorical labels, $l : V \cup E \to L$ a labeling function, $W \subseteq \mathbb{R}$ the domain of the weights and $w : E \to W$ a function which assigns weights. $E(G)$ denotes the set of edges etc.

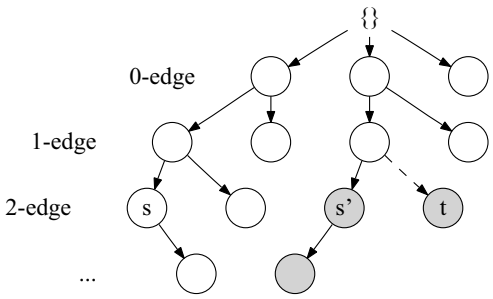


Fig. 2 A pattern-growth search space with conventional pruning (s') and constraint pruning (t).

All graphs can be directed or undirected ($e \in E$ ordered/unordered). All techniques discussed in this paper can easily be extended to cover weighted nodes ($w : V \cup E \rightarrow W$) and tuples of weights ($W \subseteq \mathbb{R}^n, n \in \mathbb{N}$).

Definition 2. *Frequent subgraph mining* is the task of finding all subgraph patterns $f \in F$ with a *support* of at least $sup_{\min}$ in a graph database $D := \{g_1, ..., g_{|D|}\}$. The support of a subgraph f is $support(f,D) := |\{g | g \in D \wedge f \subseteq g\}|$ where '$\subseteq$' denotes a subgraph-supergraph relationship. In short, $f \in F \iff support(f,D) \geq sup_{\min}$. Note that subgraph isomorphism considers labels but not weights.

Definition 3. *Closed-graph-mining* algorithms discover only subgraph patterns that are closed. A graph f is closed if no other graph pattern f' is part of the result set F which has the same support and is a proper supergraph of f ($f \subset f'$).

Closed mining algorithms produce more concise result sets and make use of pruning opportunities. *Pattern-growth-based algorithms* for (closed) frequent subgraph mining [19], such as *gSpan* [17] and *CloseGraph* [18], perform a depth-first search in a pattern-search space. Starting from some small graph pattern, they extend it by systematically adding edges and determining the support of the result. If it satisfies the $sup_{\min}$ criterion, the algorithm keeps extending it. Otherwise, the pattern is pruned, and the search backtracks.

Example 3. Figure 2 is a pattern-growth search space: The leaves cannot be extended further due to the $sup_{\min}$ criterion. s' is pruned as it is isomorphic to node s already discovered. The dashed edge with node t stands for constraint-based pruning, which we introduce in the following.

Definition 4. A constraint c in *constraint-based mining* is a Boolean predicate which any $f \in F$ must fulfill. Formally, in constraint-based frequent subgraph mining, $f \in F \iff (support(f,D) \geq sup_{\min} \wedge c(f) = true)$.

Constraint predicates can be categorized into several classes. Most important is anti-monotonicity, as such constraints allow for effective pruning.

Definition 5. c is *anti-monotone* $\iff (\forall f' \subseteq f : c(f) = true \Rightarrow c(f') = true)$

A prominent example of anti-monotone constraints is the frequency criterion: If a graph has a support of at least $sup_{\min}$, all its subgraphs have the same or a larger support. Most constraints based on weights are not anti-monotone [14].

3 Weight-Based Constraints

In this section, we define the weight-based constraints we investigate in this paper. We do not deal with anti-monotone constraints, since we are interested in investigating approximate mining results from non-anti-monotone constraints. However, the techniques would work with anti-monotone constraints as well.

Definition 6. A *weight-based measure* is a function $E(p) \rightarrow \mathbb{R}$ which assigns every edge of a graph pattern p a numerical value. The function takes the weights of the corresponding edges in all embeddings of p in all graphs in D into account.

Depending on the actual problem, one can assign some numerical or categorical value (i.e., a class label) to each graph. Measures like *InfoGain* and *PMCC* make use of such values, in addition to the weights. We discuss example measures later. – If labels are not unique, subgraphs can be embedded at several positions within a graph. We consider every single embedding of a subgraph to calculate the measures.

Definition 7. A *lower bound predicate* c_l and an *upper bound predicate* c_u for a pattern p are predicates with the following structure:

$$c_l(p) := \exists\, e_1 \in E(p) : \mathit{measure}(e_1) > t_\mathrm{l};\ c_u(p) := \nexists\, e_2 \in E(p) : \mathit{measure}(e_2) > t_\mathrm{u}$$

A *weight-based constraint* is a set of c_l, c_u, or both, connected conjunctively.

The lower- and upper-bound predicates let the user specify a minimum and maximum interestingness based on the *measure* chosen. We comment on the two predicates in Section 4. Note that Definition 7 requires to consider *all* edges of a pattern p. This is necessary, as illustrated in Example 1. The value of the measure of *any* edge of p can change when the set of graphs supporting p changes.

Weight-Based Measures. Any function on a set of numbers can be used as a measure. We have chosen to evaluate three measures with a high relevance in real data-analysis problems – none of them is anti-monotone. Two of them, *InfoGain* and *PMCC*, require the existence of a class associated with each graph. Such classes are available, e.g., in any graph-classification task, and the goal of the mining process is to derive subgraph patterns for a good discrimination between the classes. We also use the *variance*, which does not depend on any class. It is useful in explorative mining scenarios where one is interested in subgraphs with varying weights.

Example 4. If one wants to search for patterns p with a certain minimum *variance* of weights, one would specify the measure '*variance*' and the threshold value t_l. The constraint then is '$\exists\, e : \mathit{variance}(e) > t_\mathrm{l}$'. This could be useful when analyzing logistics data, where one wants to find subgraphs with unbalanced load or highly varying transportation times.

Besides the measures described in the following, many further measures from data analysis can be used similarly to build weight-based constraints. This includes, say, attribute-selection measures known from decision-tree induction [12, 15].

Information Gain. The *InfoGain* [12], is a measure based on *entropy*. It is a value between 0 and 1 and quantifies the ability of an attribute A to discriminate between classes in a dataset (without a restriction to binary classes). It is frequently used in decision-tree induction and feature selection [12, 15]. In the context of weighted graphs, A refers to the weights of a certain edge of a subgraph pattern in all embeddings in all graphs in the graph database D.
Pearson's Product-Moment Correlation Coefficient (*PMCC*). The *correlation coefficient* is widely used to quantify the strength of the linear dependence between two variables. See [15] for a definition. In our graph-mining context, these two variables are the weight of a certain edge in a subgraph pattern in all embeddings in graphs in D and their binary classes. For our purposes, positive and negative correlation have the same importance, and we use the absolute value. Then *PMCC* is a value between 0 and 1 as well.
Variance. The *variance* quantifies the variation of the values of a random variable Y. It is a positive value without upper bound. In our scenarios, Y is the set of weights of a certain edge in all subgraph patterns in all embeddings in D.

4 Weight-Based Mining

We now describe how to integrate *weight-based constraints* into *pattern-growth-based frequent subgraph mining*. We first focus on vanilla pattern-growth algorithms before turning to closed mining. The basic idea is to use weight-based constraints – even if they are not anti-monotone – to prune the search space.

Example 5. Figure 2 illustrates pattern-growth mining with and without weight-based constraints. Without such constraints, s' and its successors are pruned, as s' is isomorphic to s. With weight-based constraints, the search is additionally pruned at pattern t. The dashed edge extends its parent, and t including the new edge violates a weight-based constraint. Note that it is not necessarily the newly added edge itself which violates the constraint, but any edge in t.

In concrete terms, we treat the lower and upper-bound predicates c_l and c_u (as defined in Definition 7) in weight-constraint-based mining as follows:

Approach. When a pattern p does not satisfy c_l or c_u, the search is pruned. If it is c_u that is not satisfied, p is added to the mining result, otherwise not.

The rationale behind an *upper bound* is to speed up mining by pruning the search when a sufficiently interesting edge weight is found. Therefore, we use it to prune the search, but save the current pattern. For example, if the user wants to use the graph patterns mined for classification, a pattern with one edge with a very discriminative weight will be fair enough. Clearly, larger graphs can still be more discriminative. Setting the threshold therefore involves a trade-off between efficient pruning and finding relevant graphs. Section 6 will show that small changes in the upper

Algorithm 1 *pattern-growth*$(p, D, sup_{\min}, t_l, t_u, F)$

Input: current pattern p, database D, $sup_{\min}$, parameters *measure*, t_l and t_u. **Output:** result set F.

```
 1: if p ∈ F then
 2:     return
 3: calculate weight-based measures for all edges
 4: if ∃ e1 : measure(e1) > tl ∧ ∄ e2 : measure(e2) > tu then
 5:     if (algorithm ≠ CloseGraph ∨ p is closed) then
 6:         F ← F ∪ {p}
 7:     P ← extend-by-one-edge(p, D, sup_min)
 8:     for all p' ∈ P do
 9:         pattern-growth(p', D, sup_min, tl, tu, F)
10: else
11:     if ∃ e : measure(e) > tu then
12:         F ← F ∪ {p}
13: return
```

bound do not change the results significantly. It is therefore sufficient to rely on few different threshold values to obtain satisfactory results. With a *lower bound*, the user specifies a minimal interestingness. This bound stops mining when the value specified is not reached. The rationale is that one does not expect to find any patterns which are more interesting. Note that this might miss patterns, too.

Pattern-Growth Algorithms. Algorithm 1 describes the integration into pattern-growth-based frequent subgraph mining algorithms such as *gSpan* [17]. The algorithm works recursively, and the steps in the algorithm are executed for every node in Figure 2. Lines 1–2, 6–9 and 13 are the generic steps in pattern-growth-based graph mining [19]. They perform the isomorphism test (Lines 1–2), add patterns to the result set (Line 6) and extend the current pattern (Line 7), leading to a set of frequent patterns P. The algorithm then processes them recursively (Lines 8–9) and stops depth-first search when P is empty (Line 13).

Lines 3–4 and 10–12 are new in our extension. Instead of directly adding the current pattern into the result set, the algorithm first calculates the weight-based measures (Line 3). Line 4 checks the constraints (if c_l or c_u is not set, the thresholds are zero or ∞, respectively; cf. Definition 7). If they are not violated, or the minimum size is not reached, the algorithm saves the pattern to the result set (Line 6) and continues as in generic pattern growth (Lines 8–9). Otherwise, the algorithm prunes the search, i.e., it does not continue the search in that branch. Note that this step is critical, as it determines both the speedup and the result quality. As mentioned before, we always save the last pattern before we prune due to upper bounds (Lines 11–12). This leads to result sets which are larger than those from standard graph mining when the constraints are applied in a postprocessing step.

One can realize constraints on more than one measure in the same way, by evaluating several constraints instead of one, at the same step of the algorithm. As mentioned before, mining with weight-based constraints produces a result set with unweighted subgraph patterns. In case one needs weighted subgraphs in the result set, arbitrary functions, e.g., the average, can be used to derive weights from the supporting graphs in the graph database.

Closed Mining. Closed mining returns closed graph patterns only. When dealing with weight-based constraints, we deviate from this characteristic. We favor graphs which are interesting (according to the measures) over graphs which are closed. This is because the weight-based constraints might stop mining when 'interesting enough' patterns are found. Extending the *CloseGraph* [18] algorithm is slightly more complicated than pattern growth as described before. *CloseGraph* performs further tests in order to check for closedness (Line 5 in Algorithm 1). In our extension, these tests are done after weight-based pruning. Therefore, when the search is pruned due to a constraint, it might happen that the algorithm misses a larger closed pattern. In this case it adds patterns to the result set which are not closed.
Implementation. The extensions we describe here are compatible with any pattern-growth graph miner. We for our part use the *ParSeMiS* graph-mining suite [11] with its *gSpan* [17] and *CloseGraph* [18] implementations.[1]

5 Weighted Graph Mining Applied

We now say how to exploit the information contained in the weights of graphs in different application scenarios building on weight-constraint-based frequent subgraph mining. (See [2] for details on weighted graph classification.)
Software-Defect Localization. The purpose of defect localization is to help software developers debugging programs. In our case, the result is a list of suspicious methods, sorted by their likelihood to contain a defect. A developer can then inspect the code starting with the top-ranked method. Our approach builds on the comparison of *weighted call graphs* (cf. Figure 1), representing different executions of the same program. Every graph is labeled as *failing* or *correct*, depending on whether the program execution has returned a false or a correct result. The method described in the following can detect defects of a frequently occurring category: defects leading to infections that influence the control structure of a program, i.e., those changing the call-graph structure or a call frequency (an edge weight).

In order to obtain the likelihood of a method to contain a defect, we look at two kinds of evidence: weight-based measures and subgraph structures. Firstly, we consider the measure of edges, computed during constraint-based mining. In our implementation, a method (a node) inherits the normalized maximum value from all outgoing edges in all patterns in the result as its weight-based likelihood:

$$P_{\mathrm{w}}(m) := \mathit{normalize}(\max(\mathit{measure}(\{(m,x)|(m,x) \in \mathscr{E} \wedge x \in \mathscr{V}\})))$$

where $\mathscr{V}$ and $\mathscr{E}$ are the unions of the vertex and edge sets of all subgraph patterns in the result set, and *measure* applied to a set calculates the *measure* of every element separately. We use the maximum, as one node might have many outgoing edges which are not related to the defect at all. From preliminary experiments with

[1] We provide our extensions at `sdqweb.ipd.kit.edu/wiki/ParSeMiS-Extensions/`.

different measures applied in a postprocessing setting, we know that entropy-based measures such as *InfoGain* are best suited for defect localization.

Secondly, we look at the subgraph structures. The result sets mined with weight-based constraints let us define another likelihood based on support. They contain a higher number of interesting graphs with interesting edges (according to the measure chosen) than a result set from vanilla graph mining. Therefore, it seems promising not only to give a high likelihood to edges with interesting weights. We additionally consider nodes (methods) occurring frequently in the graph patterns in the result set. We calculate this structural likelihood similar to a support value:

$$P_s(m) := |\{f | f \in F \wedge m \in f\}| / |F|$$

The next step is to combine the two likelihoods. We do this by averaging the normalized values. Preliminary experiments have shown that using these two kinds of evidence yields a more precise localization of defects.

Finding out how well defect-localization techniques perform requires an evaluation measure. Defining such a measure is not difficult, as we, the experimentators, know the defects. We use the position of the real defect in the generated list of suspicious methods. This position quantifies the number of methods to look into in order to find the defect.

Explorative Mining. Besides automated analysis steps following graph mining, another important application is explorative mining. Here, the results are interpreted directly by humans. One is interested in deriving useful information from a dataset. In our weight-constraint-based scenario, such information is represented as subgraphs with certain edge-weight properties in line with the constraints. For instance, the logistics dataset is well suited for explorative mining: As in Example 4, one might be interested in subgraphs featuring edges with high or low variance.

Evaluation in this context is difficult, as information is mined for humans. Thus, it is hard to define an evaluation measure. In this study, we focus on basic properties of the dataset mined, in particular the size of the subgraphs. The size can be seen as a measure of expressiveness, as larger subgraphs tend to be more significant.

6 Experimental Evaluation

We now investigate the characteristics of pruning with non-anti-monotone constraints, given several real-world analysis problems. We do so by comparing different application-specific quality criteria with the speedup in runtime as well as by assessing the completeness of approximate result sets. While other solutions to the real-world problems (without weighted graph mining) might be conceivable as well, studying them is not the concern of this article. Furthermore, we do not aim at demonstrating that the analysis of weighted graphs is beneficial compared to non-weighted graphs. Other studies have shown the adequateness of weighted graphs for analysis problems in various domains [1, 5, 6, 9].

Software-Defect Localization Dataset. We investigate the dataset from [1], which consists of 14 sets of 100 classified *weighted call graphs*.[2] The graphs are quite homogeneous; the following describes one of the 14 sets. The mean number of nodes is 19.6 (standard deviation $\sigma = 1.9$), the mean number of edges is 23.8 ($\sigma = 4.6$); the edge weights are quite diverse with a mean value of 227.6 ($\sigma = 434.5$).
Logistics Dataset. This dataset is the one from [6]. It is origin-destination data from a logistics company. The graphs are as follows: Nodes represent locations, edges represent transports, and the graphs are classified. We use the time needed to get from origin to destination as edge weight. See [2] for more details.
Experimental Settings. In our experiments we compare a regular *CloseGraph* implementation to ours with weight-based constraints. In preliminary experiments, *CloseGraph* performed much better than *gSpan* with our datasets while generating subgraphs of the same predictive quality. We evaluate the quality of the results with scenario-specific evaluation measures (cf. Section 5) along with the runtime. We use a single core of an AMD Opteron 2218 with 2.6 GHz and 8 GB RAM for all experiments. We mine with a $sup_{\min}$ of 3% in all experiments with the defect-localization dataset and with a $sup_{\min}$ of 16% in the logistics data.

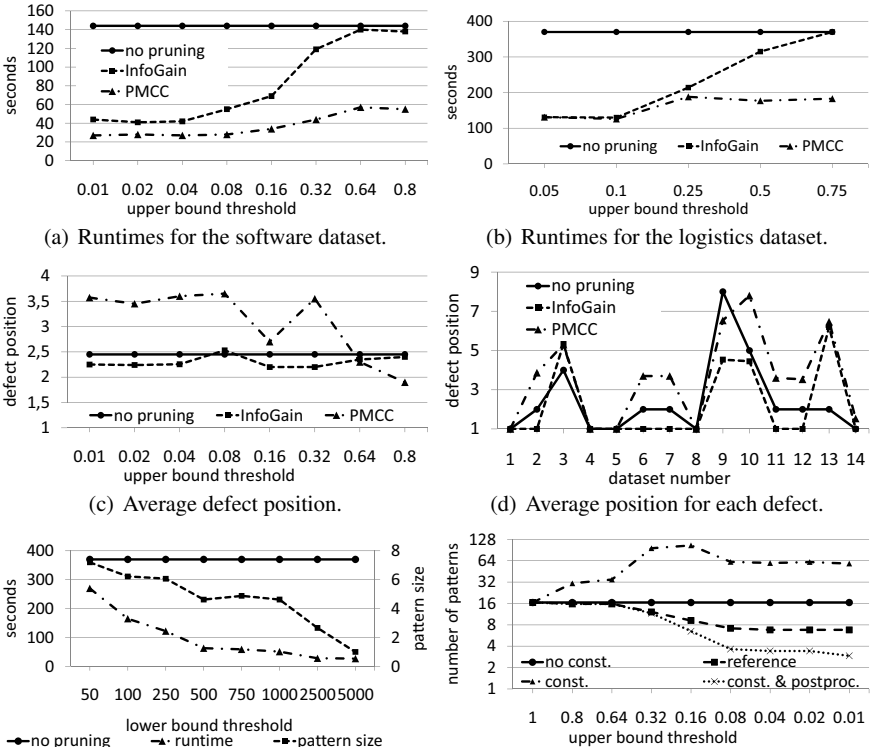
In the software-defect localization scenario, we compare our results based on edge-weight-based pruning with a vanilla graph-mining technique. To be fair, we repeat the experiments from [1] with slight revisions[3] and the same $sup_{\min}$ (3%). We use upper-bound constraints on the two class-aware measures.

For explorative-mining experiments, we investigate different lower-bound-constraint thresholds on *variance* in the logistics dataset. We compare their quality and runtime with mining runs without constraints.
Experimental Results in Software-Defect Localization. Figure 3(a) displays the runtimes of *InfoGain* and *PMCC* with different upper-bound thresholds on all 14 versions of the dataset. The *InfoGain* constraint is always faster than the execution time without pruning, irrespective of the threshold. For low threshold values (0.01 to 0.04), *InfoGain* reaches speedups of around 3.5 times. *PMCC* in turn always performs better than *InfoGain*, and reaches speedups of up to 5.2 times. This is natural, as the calculations to be done during mining in order to derive the measures are more complicated for *InfoGain* (involving logarithms) than for *PMCC*. For high thresholds (0.32 to 0.8) on both measures, the runtime increases significantly. This is caused by less pruning with such thresholds.

Figure 3(c) contains the results in defect localization without pruning and with *InfoGain* and *PMCC* pruning with various upper bounds. The figure shows the average position of the defect in the returned ranking of suspicious methods, averaged for all 14 versions. The *InfoGain* almost always performs a little bit (a fifth ranking position for the two lowest thresholds) better than the baseline ('no pruning'). As the baseline approach uses *InfoGain* as well, we explain this effect by the improved structural likelihood computation (P_s, cf. Section 5), which takes advantage of the edge-weight-based pruning. The *PMCC* curve is worse in most situations.

[2] We provide them online: `www.ipd.kit.edu/~eichi/papers/eichinger10on/`

[3] In [1], a zero in the feature vectors indicates that a certain call does not occur. We now use null values, as this allows for a fair comparison to our new approach.



(a) Runtimes for the software dataset.

(b) Runtimes for the logistics dataset.

(c) Average defect position.

(d) Average position for each defect.

(e) Runtimes & quality, logistics dataset.

(f) Comparison of approximate result sets.

Fig. 3 Experimental results.

This is as expected, as we know that entropy-based measures perform well in defect localization (cf. Section 5). Figure 3(d) contains the defect-localization results for the 14 different versions. We use the average of the three executions with the best runtime. The figure reveals that the precision of localizations varies for the different defects, and the curve representing the *InfoGain* pruning is best in all but two cases. Using the results of these mining experiments in a classification scenario, the picture is similar to the defect localization results (for details see [2]). Concerning the threshold values, observe that small changes always lead to very small changes in the resulting defect-localization precision, with mild effects on runtime.

Experimental Results in Logistics. Figure 3(b) shows the runtimes as before. With an upper bound of up to 0.10 on *InfoGain* or *PMCC*, our extension runs about 2.9 times faster than the reference without pruning. The accuracy of classification building on the results is again stable, see [2] for more details.

We also evaluate the *variance* measure in an explorative mining setting on the logistics dataset. Figure 3(e) shows the runtimes with several lower bounds along with the corresponding averaged subgraph-pattern sizes (in edges) in the result set. At the lowest threshold (50), the runtime already decreases to 73% of the runtime without

pruning. At the highest value (5,000), the runtime decreases to 7% only, which is a speedup of 13 times. At the same time, the average subgraph size decreases from 7 to 1. Therefore, values between 250 and 1,000 might be good choices for this dataset (depending on the user requirements), as the runtime is 3 to 7 times faster, while the average subgraph size decreases moderately from 7.4 to 6.1 and 4.6.

Completeness of Approximate Result Sets. We now investigate the completeness of our result sets and look at the defect-localization experiments with *InfoGain*-constraints another time. Figure 3(f) refers to these experiments with the approximate constraint-based *CloseGraph* algorithm, but displays the sizes of result sets (averaged for all 14 versions). We compare these results with a non-approximate reference, obtained from a non-constrained execution, where we remove all subgraph patterns violating an upper bound afterwards. Our constraint-based mining algorithms save all patterns violating upper bounds before pruning the search (cf. Section 4). For comparison, we apply the same postprocessing as with the reference and present two variants of constraint-based mining: The pure variant ('const.') and the postprocessed one ('const. & postproc.'). Comparing the two postprocessed curves, for thresholds of 0.64 and larger, constraint-based result sets have the same size as the reference and are smaller for thresholds of 0.32 and lower. Preliminary experiments with different $sup_{\min}$ values have revealed that the difference between the curves decreases ($sup_{\min}$ of around 20% instead of 3%) or vanishes ($sup_{\min}$ of 70%). The pure result sets (those we used in the experiments before), are always larger than closed mining, even if no constraints are applied. To conclude, our approximate result sets contain less than half of the patterns as the non-approximate reference, for small $sup_{\min}$ and upper bound values. However, the pure result sets obtained from constraint-based mining in a shorter runtime (cf. Figure 3(a)) contain many more interesting subgraph patterns (see curve 'const.').

7 Related Work

Weighted-Graph Mining. Even though weighted graphs are ubiquitous, we are only aware of a few studies analyzing such graphs with frequent subgraph mining.

[6] considers weighted logistic networks. To analyze them, the authors apply a simple discretization scheme during preprocessing. As mentioned in the introduction, this may curb result accuracy: Besides interval borders chosen deliberately, categorical intervals loose ordinal information. [9] studies image-analysis problems. The authors discretize the weights too, but more sophisticated: They cluster vectors of weights, resulting in categorical edge labels. However, the risk of loosing potentially important information by discretization persists.

In [1], we have proposed a postprocessing approach. We use classified and weighted call graphs for software-defect localization. After a graph-mining step, we use feature selection on edge weights to obtain an ordered list of edges representing potentially defective code. This approach avoids the shortcomings of discretization and analyzes numerical weights instead of discrete intervals. However, the two steps

of graph mining and feature selection are executed sequentially. This gives way to further speedups. We have investigated such an approach in this paper.

[5] formulates a text-classification task as a weighted-graph-mining problem and introduces the concept of *weighted support*. The authors achieve well results in different applications. However, mining for *weighted frequent subgraphs* offers less flexibility than constraints on arbitrary measures as investigated in this paper.

Constraint-Based Mining. [8] has introduced constraints for frequent itemset mining. The authors define the two constraint properties *anti-monotonicity* (cf. Section 2) and *succinctness*. Both help to speed up mining. [10] has introduced *convertible constraints* for itemsets, focusing on aggregate constraints.

[14] studies constraint-based graph mining. The authors extend the constraint classes introduced in [8] and integrate the different constraints into a pattern-growth graph-mining algorithm. They also propose a way to deal with some weight-based constraints. However, this can lead to incomplete result sets. Furthermore, situations where such constraints lead to significant speedups are rare, according to the authors, and they do not make any statements regarding result quality. [20] extends [14], but the study does not consider weights.

Although the techniques proposed work well with certain constraints, most weight-based constraints (cf. Section 3) do not fall into the respective categories [14]. (See Example 1 for an illustration.) They are in particular not *convertible*. The weights considered in convertible constraints [10] stay the same for every item in all transactions, while weights in graphs can be different in every graph.

Mining Significant Graph Patterns. In many settings, frequent subgraph mining is followed by a feature-selection step. This is to ease subsequent processes such as classification. As mining huge amounts of patterns and selecting a few of them in a subsequent step is expensive [16], recent studies investigate scalable algorithms [3, 7, 13, 16]. They deal with the direct mining of patterns satisfying an objective function. Depending on this function, the subgraph sets mined might be incomplete with regard to the frequency criterion, but contain all (or most) graphs with regard to the objective function chosen. Objective functions are either based on their ability to discriminate between classes or numerical values associated with the graphs [7, 13] or on some topological similarity measures [3, 16]. However, none of these approaches has taken weights into account. In our work, we use edge weights to decide which graphs are significant.

8 Conclusions

In this paper we have dealt with mining of weighted graphs, which are ubiquitous in the real world. The analysis of weights in addition to the graph structure bears the potential of more precise mining results. We have integrated non-anti-monotone constraints based on weights into pattern-growth frequent subgraph mining algorithms. This leads to improved runtime and approximate results. The goal of our study was to investigate the quality of these results. Besides an assessment of result

completeness, we have evaluated its usefulness, i.e., the result quality of higher-level real-world analysis problems based on this data.

Our study shows that a correlation of weights with the graph structure exists and can be exploited. Frequent subgraph mining with weight-based constraints has proven to be useful – at least for the problems investigated. With the software dataset, we have obtained speedups of 3.5 times. This allows for analyses of larger software projects. At the same time, the results in defect localization even are a little more precise. In the logistics dataset, we have achieved a speedup of 2.9 times. In explorative mining, the speedup is around 7 times while obtaining good results.

References

1. Eichinger, F., Böhm, K., Huber, M.: Mining Edge-Weighted Call Graphs to Localise Software Bugs. In: ECML PKDD (2008)
2. Eichinger, F., Huber, M., Böhm, K.: On the Usefulness of Weight-Based Constraints in Frequent Subgraph Mining. Tech. Rep. 2010-10, Faculty of Informatics, Karlsruhe Institute of Technology. digbib.ubka.uni-karlsruhe.de/volltexte/1000017769
3. Hasan, M.A., Chaoji, V., Salem, S., Besson, J., Zaki, M.J.: ORIGAMI: Mining Representative Orthogonal Graph Patterns. In: ICDM (2007)
4. Inokuchi, A., Washio, T., Motoda, H.: Complete Mining of Frequent Patterns from Graphs: Mining Graph Data. Mach. Learn. **50**(3), 321–354 (2003)
5. Jiang, C., Coenen, F., Sanderson, R., Zito, M.: Text Classification using Graph Mining-based Feature Extraction. Knowl.-Based Syst. **23**(4), 302–308 (2010)
6. Jiang, W., Vaidya, J., Balaporia, Z., Clifton, C., Banich, B.: Knowledge Discovery from Transportation Network Data. In: ICDE (2005)
7. Kudo, T., Maeda, E., Matsumoto, Y.: An Application of Boosting to Graph Classification. In: NIPS (2004)
8. Ng, R.T., Lakshmanan, L.V.S., Han, J., Pang, A.: Exploratory Mining and Pruning Optimizations of Constrained Associations Rules. In: SIGMOD (1998)
9. Nowozin, S., Tsuda, K., Uno, T., Kudo, T., Bakir, G.: Weighted Substructure Mining for Image Analysis. In: Conf. on Computer Vision and Pattern Recognition (CVPR) (2007)
10. Pei, J., Han, J., Lakshmanan, L.V.S.: Pushing Convertible Constraints in Frequent Itemset Mining. Data Min. Knowl. Discov. **8**(3), 227–252 (2004)
11. Philippsen, M., et al.: ParSeMiS: The Parallel and Sequential Mining Suite. Available at www2.informatik.uni-erlangen.de/EN/research/ParSeMiS/
12. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
13. Saigo, H., Krämer, N., Tsuda, K.: Partial Least Squares Regression for Graph Mining. In: KDD (2008)
14. Wang, C., Zhu, Y., Wu, T., Wang, W., Shi, B.: Constraint-Based Graph Mining in Large Database. In: Asia-Pacific Web Conf. (APWeb) (2005)
15. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (2005)
16. Yan, X., Cheng, H., Han, J., Yu, P.S.: Mining Significant Graph Patterns by Leap Search. In: SIGMOD (2008)
17. Yan, X., Han, J.: gSpan: Graph-Based Substructure Pattern Mining. In: ICDM (2002)
18. Yan, X., Han, J.: CloseGraph: Mining Closed Frequent Graph Patterns. In: KDD (2003)
19. Yan, X., Han, J.: Discovery of Frequent Substructures. In: D.J. Cook, L.B. Holder (eds.) Mining Graph Data, chap. 5, pp. 99–115. Wiley (2006)
20. Zhu, F., Yan, X., Han, J., Yu, P.S.: gPrune: A Constraint Pushing Framework for Graph Pattern Mining. In: PAKDD (2007)

Induction of Modular Classification Rules: Using Jmax-pruning

Frederic Stahl and Max Bramer

Abstract The Prism family of algorithms induces modular classification rules which, in contrast to decision tree induction algorithms, do not necessarily fit together into a decision tree structure. Classifiers induced by Prism algorithms achieve a comparable accuracy compared with decision trees and in some cases even outperform decision trees. Both kinds of algorithms tend to overfit on large and noisy datasets and this has led to the development of pruning methods. Pruning methods use various metrics to truncate decision trees or to eliminate whole rules or single rule terms from a Prism rule set. For decision trees many pre-pruning and post-pruning methods exist, however for Prism algorithms only one pre-pruning method has been developed, *J-pruning*. Recent work with Prism algorithms examined *J-pruning* in the context of very large datasets and found that the current method does not use its full potential. This paper revisits the *J-pruning* method for the Prism family of algorithms and develops a new pruning method *Jmax-pruning*, discusses it in theoretical terms and evaluates it empirically.

1 Introduction

Classification rule induction from large training samples has a growing commercial importance and can be traced back to the 1960s [7]. Two general approaches to classification rule induction exist the 'separate and conquer' and the 'divide and conquer' approaches [14]. 'Divide and conquer' is better known as Top Down Induction of Decision Trees (TDIDT) [10] as it induces classification rules in the intermediate representation of a decision tree. The 'separate and conquer' approach can be traced back to the AQ learning system in the late 1960s [9]. Compared with TDIDT AQ generates a set of *IF..THEN* rules rather than decision trees, which is

Frederic Stahl, Max Bramer
University of Portsmouth, School of Computing, Buckingham Building, Lion Terrace, PO1 3HE Portsmouth, UK e-mail: {Frederic.Stahl; Max.Bramer}@port.ac.uk

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_6,

useful for expert systems applications that are based on production rules. However most research concentrates on the TDIDT approach.

An important development of the 'separate and conquer' approach is the Prism family of algorithms [5, 2, 3]. Prism induces rules that are modular and that do not necessarily fit into a decision tree. Prism achieves a comparable classification accuracy compared with TDIDT and in some cases even outperforms TDIDT [2], especially if the training data is noisy. Recent research on the Prism family of algorithms comprises a framework that allows the parallelisation of any algorithm of the Prism family in order to make Prism algorithms scale better to large training data. The framework is called Parallel Modular Classification Rule Inducer (PMCRI) [13].

Like any classification rule induction algorithm Prism suffers from *overfitting* rules to the training data. Overfitting can result in a low predictive accuracy on previously unseen data instances (the test set) and a high number of induced rules and rule terms. There exist a variety of pruning methods for decision trees [6] that aim to reduce the unwanted *overfitting*, however there is only one published method of pruning rules of the Prism family, *J-pruning* [3]. *J-pruning* uses the *J-measure*, an information theoretic means to quantify the information content of a rule. *J-pruning pre-prunes* the rules during their induction. *J-pruning* has been integrated in PMCRI and not only improves the predictive accuracy but also lowers the number of rules and rule terms induced and thus also improves the computational efficiency of Prism algorithms [12].

This paper revisits the J-measure and *J-pruning*, develops *Jmax-pruning*, a variation of *J-pruning* and evaluates them empirically. Section 2 outlines the Prism Family of algorithms and compares them to TDIDT. Section 3 outlines *Jmax-pruning* followed by an empirical evaluation in Section 4. Ongoing work is discussed in Section 5 which comprises a new variation of the Prism approach and *Jmax-pruning* for TDIDT. Some concluding remarks can be found in Section 6.

2 The Prism Family of Algorithms

As mentioned in Section 1, rule representation differs between the 'divide and conquer' and 'separate and conquer' approaches. The rule sets generated by the 'divide and conquer' approach are in the form of decision trees whereas rules generated by the 'separate and conquer' approach are modular. Modular rules do not necessarily fit into a decision tree and normally do not. The rule representation of decision trees is the main drawback of the 'divide and conquer' approach, for example rules such as:

IF a = 1 AND b = 1 THEN class = 1

IF c = 1 AND d = 1 THEN class = 0

cannot be represented in a tree structure as they have no attribute in common. Forcing these rules into a tree will require the introduction of additional rule terms

that are logically redundant, and thus result in unnecessarily large and confusing trees [5]. This is also known as the replicated subtree problem [14].

'Separate and conquer' algorithms induce directly sets of 'modular' rules like those above avoiding unnecessarily redundant rule terms that are induced just for the representation in a tree structure. The basic 'separate and conquer' approach can be described as follows:

```
Rule_Set = [];
While Stopping Criterion not satisfied{
    Rule = Learn_Rule;
    Remove all data instances covered from Rule;
}
```

The *Learn_Rule* procedure generates the best rule for the current subset of the training data where best is defined by a particular heuristic that may vary from algorithm to algorithm. The stopping criterion is also dependent on the algorithm used. After inducing a rule, the rule is added to the rule set and all instances that are covered by the rule are deleted and a new rule is induced on the remaining training instances.

In Prism each rule is generated for a particular Target Class (TC). The heuristic Prism uses in order to specialise a rule is the probability with which the rule covers the TC in the current subset of the training data. The stopping criterion is fulfilled as soon as there are no training instances left that are associated with the TC.

Cendrowska's original Prism algorithm selects one class as the TC at the beginning and induces all rules for that class. It then selects the next class as TC and resets the whole training data to its original size and induces all rules for the next TC. This is repeated until all classes have been selected as TC. Variations exist such as PrismTC [4] and PrismTCS (Target Class Smallest first) [3]. Both select the TC anew after each rule induced. PrismTC always uses the majority class and PrismTCS uses the minority class. Both variations introduce an order in which the rules are induced, where there is none in the basic Prism approach. However the predictive accuracy of PrismTC cannot compete with that of Prism and PrismTCS (personal communication). PrismTCS does not reset the dataset to its original size and thus is faster than Prism, which produces a high classification accuracy and also sets an order in which the rules should be applied to the test set.

The basic PrismTCS algorithm is outlined below where A_x is a possible attribute value pair and D is the training dataset:

```
Step 1: Find class i that has the fewest instances in the training
        set
Step 2: Calculate  for each Ax  p(class = i| Ax)
Step 3: Select the Ax with the maximum  p(class = i| Ax)
        and create a subset D' of D that comprises all instances
        that match the selected Ax.
Step 4: Repeat 2 to 3 for D' until D' only contains instances
        of classification i. The induced rule is then a
        conjunction of all selected Ax and i.
Step 5: Create a new D' that comprises all instances of D except
        those that are covered by all rules induced so far.
Step 6: IF D' is not empty repeat steps 1 to 5 until D' does not
        contain any instances of classification i.
```

We will concentrate here on the more popular PrismTCS approach but all techniques and methods outlined here can be applied to any member of the Prism family.

2.1 Dealing with Clashes

A *clash set* is a set of instances in a subset of the training set that are assigned to different classes but cannot be separated further. For example this is inevitable if two or more instances are identical except for their classification. Cendrowska's original Prism algorithm does not take into account that there may be clashes in the training data. However the Inducer software implementations of the Prism algorithms do take clashes into account [2, 4]. What happens in the case of a clash in Inducer is that all instances are treated as if they belong to the TC. [2] mentions that the best approach is to check if the TC is also the majority class. If it is then the rule currently being induced is taken otherwise the rule is discarded. If a clash is encountered and the majority class is not the TC then the rule is discarded and all instances in the clash set that match the TC are deleted. The reason for manipulating the clash set this way is that if the rule were discarded and the clash set kept then the same rule would be induced all over again and the same clash set would be encountered again.

2.2 Dealing with Continuous Attributes

Continuous attributes are not handled by Cendrowska's original Prism algorithm. One way to deal with continuous attributes is discretisation of the attribute values prior to the algorithm's application, for example applying ChiMerge [8] before the application of a Prism algorithm. Bramer's Inducer software [4] provides implementations of Prism algorithms that deal with continuous attributes, these are also used in all Prism implementations used in this work. Dealing with continuous attributes can be integrated in step two in the pseudo code above before the calculation of $p(class = i|A_x)$. If A_x is continuous then the training data is sorted for A_x. For example let A_x comprise the following values after sorting, -3, -4.2, 3.5, 5.5 and 10, then the data is scanned for these attribute values in either ascending or descending order. For each attribute value, for example 5.5, two tests $p(class = i \mid A_x < 5.5)$ and $p(class = i \mid A_x \geq 5.5)$ are conducted. The one with the largest conditional probability for all the values of the attribute is kept and compared with those conditional probabilities from the remaining attributes.

2.3 J-pruning

As mentioned in the introduction, classifiers are generally pruned to prevent them from overfitting. Pruning methods can be divided into two categories, *pre-pruning* and *post-pruning*. Post-pruning is applied to the classifier after it has been induced whereas pre-pruning is applied during the rule induction process. For Prism algorithms only one pruning method has been developed so far, *J-pruning* [3], a pre-pruning method based on the J-measure [11], a measure for the information content of a rule. *J-pruning* can also be applied to decision tree induction algorithms and has shown good results on both kinds of algorithms [3]. As also mentioned in the introduction, *J-pruning* has found recent popularity in [12], as it reduces the number of rules and rule terms induced considerably and thus increases the computational efficiency.

According to [11] the theoretical average information content of a rule of the form *IF Y = y THEN X = x* can be measured in bits and is denoted by *J(X,Y=y)*.

$$J(X;Y=y) = p(y) \cdot j(X;Y=y) \quad (1)$$

As shown in equation (1) $J(X;Y=y)$ is essentially a product of $p(y)$, the probability with which the left hand side of the rule will occur, and $j(X;Y=y)$ which is called the j-measure (with a lower case j) and measures the goodness-of-fit of a rule. The j-measure, also called the *cross-entropy*, defined in equation (2):

$$j(X;Y=y) = p(x \mid y) \cdot log_2\left(\frac{p(x \mid y)}{p(x)}\right) + (1 - p(x \mid y)) \cdot log_2\left(\frac{(1 - p(x \mid y))}{(1 - p(x))}\right) \quad (2)$$

For a more detailed description of the J-measure Smyth's paper [11] is recommended. Bramer's essential interpretation of the J-measure is that if a rule has a high J-value then it also is likely to have a high predictive accuracy [3]. Hence the J-value is used as an indicator of whether appending further rule terms is likely to improve a rule's predictive accuracy or lower it due to overfitting. The J-value of the rule may go up or down when appending rule terms, also it may go down and up again. However it is possible to calculate the maximum J-value that the rule with its current terms might maximally achieve if additional terms were added. This upper bound cannot of course be exceeded but its value is not necessarily achievable.

Bramer's basic *J-pruning* is applied to Prism by calculating the J-value of the rule before the induction of a new rule term and the J-value that the rule would have after a newly induced rule term is appended. If the J-value goes up then the rule term is appended. In the case where the J-value goes down, the rule term is not appended and a test is applied to determine whether the majority class of the instances that are covered by the rule is also the TC. If the majority class is the TC then the rule is truncated and kept and all instances in the current subset of the training set are treated as if all instances belong the TC. If the majority class is not the TC, then the rule is discarded and the clash resolution described in Section 2.1 is invoked.

3 Variation of J-pruning

In general there is very little work on pruning methods for the Prism family of algorithms. Bramer's *J-pruning* in the Inducer software seems to be the only pruning facility developed for Prism algorithms. This section critiques the initial *J-pruning* facility and outlines *Jmax-pruning*, a variation that makes further use of the J-measure.

3.1 Critique of J-pruning

Even though *J-pruning* described in Section 2.3 achieves good results regarding the overfitting of Prism, it does not seem to exploit the J-measure to its full potential. The reason is that even if the new rule term decreases the J-value, it is possible that the J-value increases again when adding further rule terms [12]. If the rule is truncated as soon as the J-value is decreased it may result in the opposite of overfitting, an over generalised rule with a lower predictive accuracy. The relatively good results for *J-pruning* achieved in [3] could be explained by the assumption that it does not happen very often that the J-value decreases and then increases again. However, how often this happens will be examined empirically in Section 4.

3.2 Jmax-pruning

According to [11], an upper bound for the J-measure for a rule can be calculated using equation (3):

$$J_{max} = p(y) \cdot max\{\ p(x \mid y) \cdot log_2(\frac{1}{p(x)}),\ (1 - p(x \mid y)) \cdot log_2(1 \mid 1 - p(x))\} \quad (3)$$

If the actual J-value of the rule currently being generated term by term matches the maximum possible J-value (J_{max}) it is an absolute signal to stop the induction of further rule terms.

A concrete example is used to show how the J-values of a rule can develop. The example used a dataset extracted from the UCI repository, the soybean dataset [1]. Here we induce rules using our own implementation of PrismTCS without any *J-pruning*. The original dataset has been converted to a training set and a test set where the training set comprises 80% of the data instances. The 39th rule induced is:

IF (temp = norm) AND (same-lst-sev-yrs = whole-field) AND (crop-hist = same-lst-two-yrs) THEN CLASS = frog-eye-leaf-spot

This is a perfectly reasonable rule with a J-value of 0.00578. However looking at the development of the J-values after each rule term appended draws a different picture:

First Term

IF (temp = norm) THEN CLASS = frog-eye-leaf-spot

(J-value = 0.00113, J_{max} = 0.02315)

Here the rule has J-value of 0.00113 after the first rule term has been appended. The J-value for the complete rule (0.00578) is larger than the current J-value, which is to be expected as the rule is not fully specialised yet on the TC.

Second Term

IF (temp = norm) AND (same-lst-sev-yrs = whole-field) THEN CLASS = frog-eye-leaf-spot

(J-value = 0.00032, J_{max} = 0.01157)

Now the J-value is decreased to 0.00032 and J_{max} to 0.01157. Here *J-pruning* as described in Section 2.3 would stop inducing further rule terms, the finished rule would be

IF (temp = norm) THEN CLASS = frog-eye-leaf-spot

with a J-value of 0.00113. However looking at the value of J_{max}, after the second rule term has been appended, it can be seen that it is still higher than the previous J-value for appending the first rule term. Thus it is still possible that the J-value may increase again above the so far highest J-value of 0.00113. Inducing the next rule term leads to:

Third Term

IF (temp = norm) AND (same-lst-sev-yrs = whole-field) AND (crop-hist = same-lst-two-yrs) THEN CLASS = frog-eye-leaf-spot

(J-value = 0.00578, J_{max} = 0.00578)

In this case the rule was finished after the appending of the third rule term as it only covered examples of the TC. However, the interesting part is that the J-value increased again by appending the third rule term and is in fact the highest J-value obtained. Using Bramer's original method would have truncated the rule too early leading to an overall average information content of 0.00113 instead of 0.00578. The J-value and the *Jmax* value are rounded to five digits after the decimal point and appear identical but are actually slightly different. Looking at more

digits the values are in fact for the J-value 0.005787394940853119 and for the J_{max} 0.005787395266794598. In this case no further rule terms can be added to the left-hand side of the rule as the current subset of the training set only contains instances of the TC, but if this were not the case it would still not be worthwhile to add additional terms as the J-value is so close to *Jmax*.

Overall this observation strongly suggests that pruning the rule as soon as the J-value decreases does not fully exploit the J-measure's potential. This work suggests that *J-pruning* could be improved by inducing the maximum number of possible rule terms until the current subset cannot be broken down further or the actual J-value is equal to or within a few percent of *Jmax*. As a rule is generated all the terms are labelled with the actual J-value of the rule after appending this particular rule term. The partial rule for which the largest rule J-value was calculated would then be identified and all rule terms appended afterwards truncated, with clash handling as described in Section 2.1 invoked for the truncated rule [12]. We call this new pre-pruning method *Jmax-pruning*.

4 Evaluation of Jmax-pruning

The datasets used have been retrieved from the UCI repository [1]. Each dataset is divided into a test set holding 20% of the instances and a training set holding the remaining 80% of the instances.

Table 1 shows the number of rules induced per training set and the achieved accuracy on the test set using PrismTCS with *J-pruning* as described in Section 2.3 and *Jmax-pruning* as proposed in Section 3.2.

What is also listed in Table 1 as 'J-value recovers', is the number of times the J-value decreased and eventually increased again when first fully expanding the rule and then pruning it using *Jmax-pruning*. Using the original *J-pruning* as described in Section 2.3 would not detect these J-value recoveries and lead to a rule with a lower J-value and thus lower information content than it could possibly achieve.

What can be seen is that in all cases *Jmax-pruning* performs either better than or produces the same accuracy as *J-pruning*. In fact seven times *Jmax-pruning* produced a better result than *J-pruning* and nine times it produced the same accuracy as *J-pruning*. Taking a closer look in the rule sets that have been produced in the nine cases for which the accuracies for both pruning methods are the same revealed that identical rule sets were produced in seven out of these nine cases. The two exceptions are the 'Car Evolution' and 'ecoli' datasets, however in these two exceptions the classification accuracy was the same using *Jmax-pruning* or *J-pruning*. In the cases where there are identical classifiers there were no J-value recoveries present. In Section 3.1 we stated that the good performance of *J-pruning* [3], despite its tendency to generalisation, can be explained by the fact that there are not many J-value recoveries in the datasets and thus the tendency to over generalisation is low. Looking into the last column of table 1 we can see the number of J-value recoveries. In seven cases there are none, thus there is no potential for over generalisation by using

Table 1 Comparison of *J-pruning* and *Jmax-pruning* on PrismTCS.

Dataset	Number of Rules	Accuracy (%)	Number of Rules	Accuracy (%)	J-value recovers
	J-Pruning		J-max Pruning		
monk1	4	79	12	86	4
monk3	3	98	3	98	0
vote	3	94	3	94	0
genetics	8	70	8	70	0
contact lenses	4	95	4	95	0
breast cancer	24	96	24	96	0
soybean	39	88	43	89	4
australian credit	20	89	20	89	0
diabetes	29	75	31	76	1
crx	18	83	18	83	0
segmentation	83	79	86	82	2
ecoli	23	78	26	78	3
Balance Scale	10	72	36	74	21
Car Evaluation	4	76	4	76	1
Contraceptive Method Choice	19	44	28	45	8
Optical Recognition of handwritten Digits	456	57	467	58	6

J-pruning and for the remaining datasets there is only a very small number of J-value recoveries with the exception of the 'Balanced Scale' dataset for which a 2% higher accuracy has been retrieved by using *Jmax-pruning* compared with *J-pruning*.

Loosely speaking, if there are no J-value recoveries present, then Prism algorithms with *Jmax-pruning* will produce identical classifiers to Prism algorithms with *J-pruning*. However, if there are J-value recoveries, it is likely that Prism algorithms with *Jmax-pruning* will produce classifiers that achieve a better accuracy than Prism algorithms with *J-pruning*.

What can also be read from Table 1 is the number of rules induced. In all cases in which both pruning methods produced the same accuracy, the classifiers and thus the number of rules were identical. However in the cases where the J-value recovered, then the number of rules induced with *Jmax-pruning* was larger than the number of rules induced with *J-pruning*. This can be explained by the fact that in the case of a J-value recovery the rule gets specialised further than with normal *J-pruning* by adding more rule terms while still avoiding overfitting. Adding more rule terms results in the rule covering fewer training instances from the current subset. This in turn results in that before the next iteration for the next rule less instances are

deleted from the training set, which potentially generates more rules, assuming that the larger the number of training instances the more rules are generated.

5 Ongoing Work

5.1 J-PrismTCS

Annother possible variation of PrismTCS that is currently being implemented is a version that is solely based on the J-measure. Rule terms would be induced by generating all possible categorical and continuous rule terms and selecting the one that results in the highest J-value for the current rule instead of selecting the one with the largest conditional probability. Again the same stopping criterion as for standard PrismTCS could be used, which is that all instances of the current subset of the training set belong to the same class. We call this variation of PrismTCS, *J-PrismTCS*.

5.2 Jmax-Pruning for TDIDT

J-pruning has also been integrated into the TDIDT approach as a pre-pruning facility and achieved a higher classification accuracy than TDIDT without *J-pruning* [3]. Encouraged by the good results outlined in Section 4 which were achieved with *Jmax-pruning* in PrismTCS, we are currently developing a version of *Jmax-pruning* for TDIDT algorithms. The following pseudo code describes the basic TDIDT algorithm.

```
IF   All instances in the training set belong to the
     same class
THEN return value of this class
ELSE (a) Select attribute A to split on
     (b) Divide instances in the training set
         into subsets, one for each value of A.
     (c) Return a tree with a branch for each non
         empty subset, each branch having a decendent
         subtree or a class value produced by applying
         the algorithm recursively
```

The basic approach of *J-pruning* in TDIDT is to prune a branch in the tree as soon as a node is generated at which the J-value is less than at its parent node [3]. However performing the *J-pruning* is more complicated than for Prism algorithms as illustrated in the example below.

Figure 1 illustrates a possible tree which is used to explain *J-pruning* and *Jmax-pruning* for TDIDT. The nodes labelled with '?' are placeholders for possible subtrees. Now assuming that a depth first approach is used and the current node being expanded is node 'D'. In this case *J-pruning* would take the incomplete rule

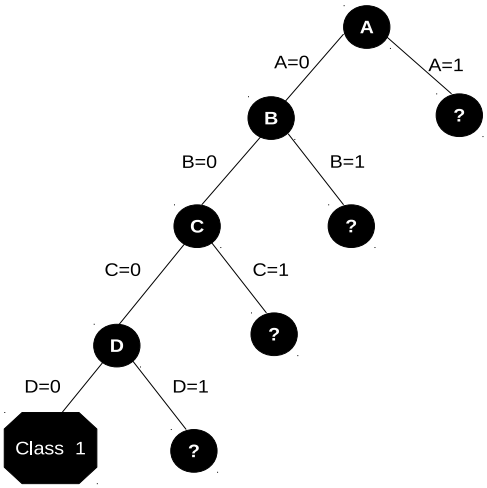


Fig. 1 Example of a decision tree for *J-pruning*.

(1) IF (A=0) AND (B=0) AND (C=0)...

the complete rule

(2) IF (A=0) AND (B=0) AND (C=0) AND (D=0) THEN class = 1

and the possible incomplete rule

(3) IF (A=0) AND (B=0) AND (C=0) AND (D=1)...

into account. Rule (2) is completed as all instances correspond to the same classification which is (class = 1). However instances covered by incomplete rule (3) correspond in this case to more than one classification.

J-pruning now compares the J-values of the incomplete rules (1) and (3). If the J-value of rule (3) is less than the J-value of rule (1) then rule (3) is completed by assigning it to the majority class of the corresponding instances. The complication is that the calculation of the J-value of a rule requires us to know its right-hand side. In the case of a complete (non-truncated) rule, such as rule (2), this is straightforward, but how can the J-value be calculated for an incomplete rule?

The method described by Bramer [3] is to imagine all possible alternative ways of completing the incomplete rule with right-hand sides class=1, class=2 etc., calculate the J-value of each such (completed) rule and take the largest of the values as the estimate of the J-value of the incomplete rule.

In a similar way to J-pruning for Prism algorithms, J-pruning for TDIDT in its current form does not necessarily exploit the full potential of the J-measure as again it is possible that if rule (3) were not truncated at the node labelled '?' but expanded to complete the decision tree in the usual way the J-value for some or possibly all of the resulting complete branches might be at least as high as the J-value at node D.

Applying the idea of Jmax-pruning rather than J-pruning to TDIDT may increase the classification accuracy. This could be done by developing the complete decision tree and labelling each internal node with a J-value estimated as described above. Each branch (corresponding to a completed rule) can then be truncated at the node that gives the highest of the estimated J-values, in a similar way to the method described in Section 3.2, with each truncated rule assigned to the majority class for the corresponding set of instances.

This method appears attractive but there is a possible problem. Using the example from figure 1 and assuming that the estimated J-value of rule (1) is greater than the estimated J-value of rule (3) and that the majority class of the instances at node D is '1', then rule (1) would be truncated at node D and rule (3) would cease to exist, giving two completed rules in this subtree:

(1) IF (A=0) AND (B=0) AND (C=0) THEN class = 1

and

(2) IF (A=0) AND (B=0) AND (C=0) AND (D=0) THEN class = 1

Both rules are illustrated in figure 2. Rule (2) is now redundant. It is just a special case of rule (1), with the same classification, and can be discarded.

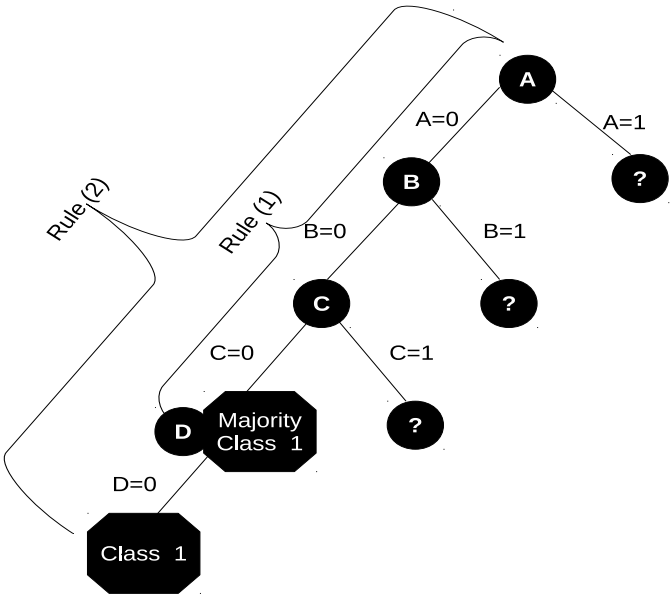


Fig. 2 Example of a decision tree with a redundant rule

Now suppose instead that in the above the majority class of the instances at node (1) were '2' (rather than '1'). In this case a different picture would emerge, with rules (1) and (2) having different classifications. How likely this situation is to occur in practice and how it should best be handled if it does both remain to be determined.

6 Conclusions

Section 2 discussed the Prism family of algorithms as an alternative approach to TDIDT to the induction of classification rules. The Prism family of algorithms was highlighted and *J-pruning*, a pre-pruning facility for Prism algorithms based on the J-measure, which describes the information content of a rule, was introduced. Section 3 criticised *J-pruning* as it does not fully exploit the potential of the J-measure. The J-value of a rule may go up or down when rule terms are appended to the rule. *J-pruning* truncates a rule as soon as the J-value decreases even if it may recover (increase again). The proposed *Jmax-pruning* exploits the possibility of a J-value recovery and achieves in some cases, examined in Section 4, better results compared with *J-pruning*, but in every case examined *Jmax-pruning* achieved at least the same or a higher classification accuracy compared with *J-pruning*.

The ongoing work comprises the development of J-PrismTCS, a version of PrismTCS that is solely based on the J-measure, by using it also as a rule term selection metric as discussed in Section 5.1. Furthermore the ongoing work comprises the development of a TDIDT algorithm that incorporates *Jmax-pruning* as discussed in Section 5.2.

References

1. C L Blake and C J Merz. UCI repository of machine learning databases. Technical report, University of California, Irvine, Department of Information and Computer Sciences, 1998.
2. M A Bramer. Automatic induction of classification rules from examples using N-Prism. In *Research and Development in Intelligent Systems XVI*, pages 99–121, Cambridge, 2000. Springer-Verlag.
3. M A Bramer. An information-theoretic approach to the pre-pruning of classification rules. In B Neumann M Musen and R Studer, editors, *Intelligent Information Processing*, pages 201–212. Kluwer, 2002.
4. M A Bramer. Inducer: a public domain workbench for data mining. *International Journal of Systems Science*, 36(14):909–919, 2005.
5. J. Cendrowska. PRISM: an algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 27(4):349–370, 1987.
6. F Esposito, D Malerba, and G Semeraro. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):476–491, 1997.
7. E B Hunt, P J Stone, and J Marin. *Experiments in induction*. Academic Press, New York, 1966.
8. R Kerber. Chimerge: Discretization of numeric attributes. In *AAAI*, pages 123–128, 1992.

9. R S Michalski. On the Quasi-Minimal solution of the general covering problem. In *Proceedings of the Fifth International Symposium on Information Processing*, pages 125–128, Bled, Yugoslavia, 1969.
10. R J Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, 1993.
11. P. Smyth and R M Goodman. An information theoretic approach to rule induction from databases. *Transactions on Knowledge and Data Engineering*, 4(4):301–316, 1992.
12. F T Stahl. *Parallel Rule Induction*. PhD thesis, University of Portsmouth, 2009.
13. F T Stahl, M A Bramer, and M Adda. PMCRI: A parallel modular classification rule induction framework. In *MLDM*, pages 148–162. Springer, 2009.
14. I H Witten and F Eibe. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

A Kolmogorov Complexity View of Analogy: From Logical Modeling to Experimentations

Meriam Bayoudh, Henri Prade, Gilles Richard

Abstract Analogical reasoning is considered as one of the main mechanisms underlying human intelligence and creativity, allowing the paradigm shift essential to a creative process. More specific is the notion of analogical proportion like "2 is to 4 as 5 is to 10" or "read is to reader as lecture is to lecturer": such statements can be precisely described within an algebraic framework. When the proportion holds between concepts as in "engine is to car as heart is to human" or "wine is to France as beer is to England", applying an algebraic framework is less straightforward and a new way to understand analogical proportions on the basis of Kolmogorov complexity theory may seem more appropriate. This viewpoint has been used to develop a classifier detecting analogies in natural language. Despite their apparent difference, it is quite clear that the two viewpoints should be strongly related. In this paper, we investigate the link between a purely abstract view of analogical proportions and a definition based on Kolmogorov complexity theory. This theory is used as a backbone to experiment a classifier of natural language analogies whose results are consistent with the abstract setting.

1 Introduction

Despite its specific status, analogical reasoning can be considered as a very common reasoning process and has the ability to shortcut long classical reasoning leading to the same conclusion. It is largely accepted that analogy is the basis for creativity as it puts different paradigms into correspondence (see [1, 2]). Analogical reasoning is

Meriam Bayoudh
IRIT, Université Paul Sabatier, 31062 Toulouse Cedex 09, France e-mail: bayoudh@irit.fr,

Henri Prade
IRIT, Université Paul Sabatier, 31062 Toulouse Cedex 09, France e-mail: prade@irit.fr,

Gilles Richard
British Institute of Technology and E-commerce, Avicenna House 258-262 Romford Road London E7 9HZ, e-mail: grichard@bite.ac.uk

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_7,

based on the human ability to identify "situations" or "problems" a and c, and then "deduce" that if b is a solution for the problem a, then some d, whose relation to c is similar to the relation between a and b, could be a solution for problem c. Such a relation involving 4 items a,b,c,d is called an analogical proportion or analogy for short, usually denoted $a : b :: c : d$ and should be read "a is to b as c is to d". Algebraic frameworks for giving concise definitions of analogical proportions have been deeply investigated in [3]. For instance, when the universe is the set $\mathbb{R}$ of real numbers, the truth of $a : b :: c : d$ is interpreted as $a \times d = b \times c$, justifying "2 is to 4 as 5 is to 10". Another example now involving sequences of bits could be *01 is to 10 as 11 is to 00* just because 01 and 10 does not share any bit and this is the case with 11 and 00. In [4, 5], a complete logical framework has been developed where a,b,c,d are described in terms of Boolean features. Besides in AI, analogy-discovering programs have been designed for specialized areas where a minimal algebraic structure is underlying the statements (see, e.g. [6]).

Natural language analogies like "engine is to car as heart is to human" or "wine is to France as beer is to England" are more at a linguistic or conceptual level: a simple mathematical structure is missing to cope with such proportions. Sowa's conceptual graphs (CG) offer an appealing framework for representing concepts: core knowledge can be encoded using CG, and then with the help of a structured linguistic database (like e.g. WorldNet), we could discover analogies as with VivoMind analogy engine [2]) for instance. In [7], a method dealing with natural language analogies but avoiding any pre-coding of the universe has been developed. The main idea is that each word a carries an "information content" that is formally defined via its Kolmogorov complexity, $K(a)$, which is an ideal natural number. In order to build up an effective implementation, this number is estimated via the World Wide Web used as a huge linguistic repository and Google as a mining engine [7]. It appears that the definitions are consistent with a sample of well-agreed analogies. Starting from these previous works, we first re-implement a classifier using a structured database coming from the US National Institute of Standards and Technology (NIST) TREC Document Databases (http://www.nist.gov/srd/nistsd22.htm). Then a careful examination of our results, which are slightly better than the ones coming from the Web, leads to propose other options, bridging the gap between a purely Boolean view of analogical proportions and a Kolmogorov complexity definition.

Our paper is organized as follows: the next section starts from an informal analysis of the core ideas underlying an analogical proportion which leads to the well agreed axioms governing this proportion. We also provide the Boolean interpretation of such a proportion and highlights the properties we expect to be satisfied in another context. In Section 3, we switch to natural language analogies, briefly recalling the main principles of Kolmogorov complexity theory and its companion notion known as the universal distribution. We show how to use it to provide different practical definitions for analogical proportions between the conceptual meaning of words, highlighting the link with the logical setting described in Section 2. In Section 4, we examine the results we get through diverse sets of experimentations and we show that they bridge the apparent gap between the Boolean framework and the complexity-based framework. Finally we survey related works and conclude.

2 Analogical proportions: a logical view

An analogical proportion[1] can be considered as a relation involving 4 items and satisfying some basic axioms which are supposed to capture its essence and that we recall below. Let us start with a brief informal analysis of the core ideas underlying this relation.

2.1 Brief analysis

Analogy puts two situations in parallel and compare them by putting them into correspondence. In a simple form, each situation involves two entities or items, say a, c on the one hand, and b, d on the other hand. The comparison then bears on the pair a and b, and on the pair c and d. This naturally lead to consider two kinds of properties:

- what is common in terms of properties to a and b: let us denote it $com(a,b)$,
- and what is specific to a and not shared by b: we denote it $spec(a,b)$.

Due to the intended meaning of com and $spec$, it is natural to assume $com(a,b) = com(b,a)$ but in general, we cannot assume $spec(a,b) = spec(b,a)$: $spec(a,b) \neq spec(b,a)$ is more realistic. With this view,

- a is represented by the pair $(com(a,b), spec(a,b))$
- b is represented by the pair $(com(a,b), spec(b,a))$

while

- c is represented by the pair $(com(c,d), spec(c,d))$
- d is represented by the pair $(com(c,d), spec(d,c))$

Then, an analogical proportion between the 4 items, expressing that *a is to b as c is to d* amounts to state that the way a and b differ is the same as the way c and d differ, namely using our notation:

$$spec(a,b) = spec(c,d) \text{ and } spec(b,a) = spec(d,c)$$

assuming symmetry in the way the parallel is done. This simple informal observation highlights two expected properties:

- *a is to b as a is to b* and
- if *a is to b as c is to d* then *c is to d as a is to b* (due to the symmetry of the $=$ operator)

We can also observe above that since $spec(a,b) = spec(c,d)$, it means that a differs from c through the properties shared with b previously denoted $com(a,b)$, and it is the same for b with respect to d. This amounts to write $spec(a,c) = spec(b,d)$, and

[1] From time to time, in the remaining of this paper the word "analogy" will be used as a shortcut for "analogical proportion".

similarly $spec(c,a) = spec(d,b)$, which together exactly mean *a is to c as b is to d*. We retrieve here the central permutation postulate that most authors associate with analogical proportion together with the symmetry postulate already mentioned. It is time now for a formalization.

2.2 Formal setting

A natural modeling option is to consider a first order setting where a,b,c,d are variables and A denotes a quaternary relation. A is an analogical proportion when it satisfies the following axioms:

- $A(a,b,a,b)$ (identity)
- $A(a,b,c,d) \implies A(c,d,a,b)$ (symmetry)
- $A(a,b,c,d) \implies A(a,c,b,d)$ (central permutation)

Using these axioms, we infer that an analogical relation A should satisfy $A(a,a,a,a)$ and $A(a,a,b,b)$ which is intuitively satisfactory but $A(a,b,b,a)$ does not hold in general as soon as $a \neq b$ (back to our informal analysis, this is due to the fact that $spec(a,b) \neq spec(b,a)$). These axioms, often mentioned, are supposed to capture the fundamental properties of analogical proportion. Clearly, the third postulate (central permutation) is the strongest one and is in some sense specific of analogical proportion. In case of analogy between numbers, a ratio-based reading is natural as for instance in the example $3:6::4:8$ and obviously agrees with the idea of central permutation. This is also the case with a difference-based reading for a numerical analogy such as $13:15::17:19$. When it comes to geometry, a,b,c and d are vectors or points in $\mathbb{R}^2$: to be in analogical proportion, they have to be the vertices of a parallelogram, $\overrightarrow{ab} = \overrightarrow{cd}$, which is equivalent to $d(a,b) = d(c,d)$ and $d(a,c) = d(b,d)$ (where d is the Euclidean distance). But, when it comes to analogical proportions between words representing concepts, it may be more problematic: general analogical statements as "engine is to car as heart is to human" or "wine is to France as beer is to England" have to be handled differently. This situation will be examined in Section 3.
Basic properties of analogy can be easily deduced from the axioms. For instance:

Proposition 1 *If A is an analogical relation, then the 6 following properties hold:*
$A(a,b,c,d) \rightarrow A(c,a,d,b)$ *(i) (by symmetry + central permutation)*
$A(a,b,c,d) \rightarrow A(b,d,a,c)$ *(ii) (by central permutation + symmetry)*
$A(a,b,c,d) \rightarrow A(b,a,d,c)$ *(iii) (by ii + central permutation)*
$A(a,b,c,d) \rightarrow A(d,c,b,a)$ *(iv) (by iii + symmetry)*
$A(a,b,c,d) \rightarrow A(d,b,c,a)$ *(v) (by iv + central permutation)*

This means that, when an analogical proportion holds for (a,b,c,d), the same proportion holds for 7 permutations of (a,b,c,d), leading to a class of 8 permutations satisfying the proportion. When there is no ambiguity about the context, the standard notation (that we use in the remaining of this paper) for the analogy $A(a,b,c,d)$ is $a:b::c:d$. Let us consider now a Boolean interpretation of analogy.

2.3 Boolean interpretation

When the items a,b,c,d belong to a structured universe, it is relatively easy to define an analogical proportion and this has been done for diverse universes: Boolean lattice, sets, strings, etc (see [3] for instance). In this section, we recall the Boolean model (where items belong to $\mathbb{B} = \{0,1\}$) as defined in [8], and we underline some remarkable properties. In that case, $a : b :: c : d$ is defined as the following Boolean formula:

$$((a \wedge \neg b) \equiv (c \wedge \neg d)) \wedge ((\neg a \wedge b) \equiv (\neg c \wedge d))$$

This formula is true for the 6 truth value assignments of a,b,c,d appearing in Table 1, and is false for the $2^4 - 6 = 10$ remaining possible assignments.

Table 1 Analogy truth table: Boolean model

a	**b**	**c**	**d**
0	0	0	0
1	1	1	1
0	1	0	1
1	0	1	0
0	0	1	1
1	1	0	0

This relation over $\mathbb{B}^4$ satisfies the 3 axioms required for an analogical proportion and several equivalent writings have been proposed in [8] e.g.

Definition 1 $(a : b :: c : d)$ *holds iff* $((a \rightarrow b) \equiv (c \rightarrow d)) \wedge ((b \rightarrow a) \equiv (d \rightarrow c)) = 1$.

This operator enjoys some properties such as transitivity that are not detailed here (see [8] for a complete review). But apart from that, and due to the existence of a negation operator in the Boolean lattice, it is interesting to examine the behavior of analogical proportion with regard to negation. It can be easily shown that the 3 following properties hold:

Proposition 2 $a : b :: c : d \rightarrow \neg a : \neg b :: \neg c : \neg d$ *(i)*
$a : b :: c : d \rightarrow \neg b : \neg a :: c : d$ *(ii)*
$a : b :: \neg b : \neg a$ *(iii)*

Property (i) expresses a kind of code-independency: an analogical proportion does not rely on the way we code the truth (1) and the falsity (0). Property (ii) expresses that we can negate one side of the :: by reversing the items. Thank to the central permutation axiom, this is equivalent to $a : b :: c : d \rightarrow c : \neg b :: d : \neg a$. Property (iii), added to the fact that neither $a : b :: \neg a : \neg b$ nor $a : \neg a :: b : \neg b$ hold, could seem counter-intuitive at first glance. Nevertheless, this should not come as a surprise if we remember that analogy is only characterized by the three axioms of Section 2 which do not constrain its behavior with respect to operators that are associated to a particular interpretative setting. In fact, the two last stated proportions hold at the level of their outlook, but not intrinsically. Indeed it can easily be checked that in

terms of descriptive features, the two analogical proportions hold in the sense of Table 1 as shown in Table 2.

Table 2 Boolean modeling for the patterns $a : b :: \neg a : \neg b$ and $a : \neg a :: b : \neg b$

	1st alph. letter	2nd alph. letter	positive		1st alph. letter	2nd alph. letter	positive
a	1	0	1	a	1	0	1
b	0	1	1	$\neg a$	1	0	0
$\neg a$	1	0	0	b	0	1	1
$\neg b$	0	1	0	$\neg b$	0	1	0

When, instead of dealing with Boolean values, we deal with concepts represented as words (like "car" or "human"), we cannot rely on any pre-existing structure to provide a definition for analogical proportion. Before leaving this section, let us investigate how we could deal with such analogies by applying algebraic methods.

2.4 Formal frameworks to cope with natural language analogies

Let us consider two typical analogies, namely "read is to reader as write is to writer" and "heart is to human as engine is to car". Back to our initial analysis, when $a : b :: c : d$ holds, it means that a and b differ in the same way as c and d differ (we use the notion of specificities).

In the case of "read is to reader as write is to writer" where a, b, c, d are easily identified, we have $spec(a,b) = \emptyset = spec(c,d)$ and $spec(b,a) =$'er'$= spec(d,c)$.

The case of "heart is to human as engine is to car" is more tricky and do not rely on a simple syntactic operation such as adding or deleting $'er'$. In fact, the words we use are here to represent concepts and implicitly they call for external pieces of knowledge such as :

$partOf(heart, human)$, $stop(heart) \rightarrow \neg move(human)$,
$stop(heart) \rightarrow \neg think(human)$,
$inFunction(human) \rightarrow move(human) \vee think(human)$

which constitutes an implicite knowledge base from which we can infer

$\neg(move(human) \vee think(human)) \rightarrow \neg inFunction(human)$

and finally $stop(heart) \rightarrow \neg inFunction(human)$.

The same kind of implicit knowledge applies to car and engine:

$partOf(engine, car)$, $stop(engine) \rightarrow \neg move(car)$,
$stop(engine) \rightarrow \neg neutralGear(car)$,
$inFunction(car) \rightarrow move(car) \vee neutralGear(car)$

from which we finally infer $stop(engine) \rightarrow \neg inFunction(car)$. Let us denote $KB(heart, human)$ the first knowledge base and $KB(engine, car)$ the second one. In some sense, $KB(heart, human)$ (resp. $KB(engine, car)$) specifies the link between *heart* and *human* (resp. *engine* and *car*). Evaluating the analogy amounts to compare and notice the (partial) identity of this link. Obviously "wine is to France as

beer is to England" would lead to the same kind of treatment but using other predicates leading for instance to:

$isaDrink(beer), isaDrink(wine), drink(wine, France), drink(beer, England)$
$alcohol(beer), alcohol(wine), isaCountry(France), isaCountry(England)$

Representing this knowledge base with a Boolean table, we get:

Table 3 Boolean modeling for "wine is to France as beer is to England"

	alcohol	**isaDrink**	**isaCountry**	**drink**(**beer**, **England**)	**drink**(**wine**, **France**)
wine	1	1	0	0	1
France	0	0	1	0	1
beer	1	1	0	1	0
England	0	0	1	1	0

We observe that the analogical proportion holds componentwise, which allows us to conclude the proportion holds on its whole. Proceeding this way, the analogical proportion becomes straightforward and only relies on atomic proportions.
As can be seen, structural information is here added to the core knowledge from which we can work, but this is not always an easy task to identify such a structure. That is why in [7], a different viewpoint has been developed that do not rely on any representation or structure and that we describe in the next section.

3 Analogies in natural language: an information-theoretic view

"Wine is to France as beer is to England" is a good example of the kind of analogical proportions we try to approach in this section. The idea is that conceptual representations would be only *implicit* and summarized in terms of information amounts, noticing that we are interested in what "information" is common to a and b (resp. c and d), and more importantly in what "information" is added/deleted when "going from" a to b or from c to d. So if we are able to properly define this notion of "information" for words representing concepts, it could be the basis for a quantitative information-based interpretation of analogy between concepts. This is why we turn to information theory and more precisely toward Kolmogorov complexity.

3.1 Kolmogorov theory: a brief overview

Developed in the late 1960's, the aim of Kolmogorov complexity theory was to give a formal meaning to the notion of 'information content'. For a given string x, Kolmogorov complexity $K(x)$ is a numerical measure of the descriptive complexity contained in x. In this paper, we simply give some notations and intuitions that are useful to understand our work. We start from a Universal Turing Machine $\mathscr{U}$, with an input tape containing a string y, a program tape containing a string p and an output tape. Universal simply means that any other machine can be simulated with $\mathscr{U}$: following Church's thesis, there are such machines. When we start to run p on

$\mathscr{U}$ with y as input, if the machine halts, we have a finite string x on the output tape and a finite part of p, pr has been read. It is convenient to adopt a functional notation with: $\mathscr{U}(y, pr) = x$. It means that there is a way to transform y into x using pr or any program with pr as prefix. Another way to put the things is to say that pr can reconstruct x with the help of auxiliary data y. Then the conditional Kolmogorov complexity of x relative to y is:

Definition 2

$$K(x|y) = min\{|pr| : \mathscr{U}(y, pr) = x\}$$

In some sense, $K(x|y)$ represents the shortest way to go from y to x. Then the Kolmogorov complexity of x is just

Definition 3

$$K(x) = K(x|\varepsilon) \textit{ where } \varepsilon \textit{ denotes the empty string.}$$

Given a program p such that $|p| = K(x)$, able to produce x from $\mathscr{U}$ with no auxiliary string, it can be understood as the essence of x since we cannot recover x from a shorter program than p. It is thus natural to consider p as the most compressed version of x and the size of p, $K(x)$, as a measure of the amount of information contained in x. With this viewpoint, $K(x|y)$ measures the amount of information we need to recover x from y. K is extended to pair of strings simply by putting that $K(x,y)$ is the length of the shortest program which can output the pair $< x, y >$ and then halt. Having a concise definition for "information content", we have the necessary tool for defining a quantitative view of analogical proportion in the context of natural language.

4 Kolmogorov model for analogy in natural language

Taking inspiration from the definitions above, [7] starts with some obvious and simple ideas extracted from the definitions:

- we work on words representing concepts. a, b, c and d are viewed as simple strings and we have only access to their information content via K.
- following our initial analysis, we maintain the common understanding that "a:b::c:d" holds inasmuch as a and b agree/disagree in the same way as c and d.

Using the Kolmogorov framework previously described, it makes sense to consider $K(b/a)$ as a measure of the quantity of information we have to handle (add or remove or transform) to go from a to b. We have now several options:

1. We can interpret analogy between concepts (represented with words) as the exact counterpart of the Boolean interpretation considering the information theoretic translation of $a \wedge \neg b$ as $K(a/b)$. Then $a : b :: c : d$ holds iff:

$$[K(a/b) = K(c/d) \wedge (K(b/a) = K(d/c))] \quad (I_1)$$

Obviously, this definition obeys the first two axioms (identity and symmetry) of an analogical proportion. But there is no way to infer the central permutation property.

2. To take into account the central permutation postulate, required for a realistic interpretation of analogy, we can enforce the fact that $a : c :: b : d$ should hold as well, then leading to define $a : b :: c : d$ as

$$(I_1) \wedge [(K(a/c) = K(b/d)) \wedge (K(c/a) = K(d/b)] \ (I_2)$$

Having such definitions to validate the formulas, it remains to estimate K. This is done through the idea of universal distribution.

Universal distribution We have to go back to the works of Solomonoff [9] whose idea was to define a kind of universal distribution over all possible objects to overcome the problem of unknown prior distribution within Bayes' formula. His idea was to considered $2^{-K(a)}$ as the unknown *apriori* probability of a when nothing else is known. In fact, in order to define a true probability measure, this definition has to be refined using specific constraints on the type of authorized programs (reduced programs) [10, 11, 12]. With this in mind, the application $a \rightarrow 2^{-K(a)}$ is a probability distribution over the set of finite strings $\{0,1\}^{\mathcal{N}}$. With our point of view, we can understand this number as the probability for a to appear (i.e. in that case, to be produced by a Turing machine). As it is quite clear that the log inverse of $2^{-K(a)}$ is just $K(a)$, any process generating strings and whose mass distribution is known can be used as a Kolmogorov complexity estimator: if $p(a)$ is the probability of a to be generated by the process, then an estimate of $K(a)$ is just $-log_2(p(a))$. It remains for us to find out a process generating a known mass distribution over words, relevant for our purpose.

5 Experimentations

In this section, we show how our different experimentations validate (at least partially) the ideas described above, using the same kind of strategy as in [13, 7] that we recall below.

5.1 Probability distribution generator

Since our words (or strings) are just syntactic representations for concepts, it is relevant to deal with a corpus of texts where these words are in use and thus get their meaning. When we are looking for a word a, querying the collections of texts to get the number n of pages where a appears, then dividing this number by the total number M of pages in the corpus, we get the frequency $p = n/M$ of this word in our collection (more precisely the frequency of the pages containing at least one occurrence of the word). Considering this frequency as a probability, and using the log inverse function, we get $-log_2(p)$ as an estimation of the Kolmogorov com-

plexity of a. But we have to deal with $K(a/b)$ as well: this is naturally estimated via $-log_2p(a|b) = -log_2p(a,b)/p(b) = log_2p(b) - log_2p(a,b)$ where $p(a,b)$ is the proportion of pages containing both the words a and b. Thus, we have everything we need for our experimentations. This work has been done with two corpora:

- in [7], with the World Wide Web, using Google as an effective tool to mine it.
- in this paper, with a TREC database (GOV) containing one year of US government proceedings ((http://www.nist.gov/srd/nistsd22.htm).

From http://www.teachersdesk.org/vocabanal.html, we get a list of well agreed analogies. To build up our negative examples, we proceed in two ways:

- method 1: starting from an analogy $a : b :: c : d$ of the previous list, we build up $a : b :: d : c$ as a negative example, switching their 2 last items in the proportion.

- method 2: starting from an analogy $a : b :: c : d$, we randomly choose a word d' without any link with a, b, c, d and we build up $a : b :: c : d'$ as a negative example.

At the end of this process, we get a testing set of 150 elements altogether.

5.2 Results

We summarize our results in terms of confusion matrix. Let us first recall the matrix we got in [7] when the WEB is used as a text corpus, Google as a querying engine and with I_2 as definition:

I_2:		positive examples	negative examples
	+	38	25
	−	12	75

This leads to an accuracy rate of 113/150 * 100 % $\simeq$ 75%. This result has been improved to go up to 80% thank to a careful examination of the failures where it appears that polysemic words (for instance "glass") cause a lot of errors: we replaced these polysemic examples with non polysemic ones, which is possible due to the large size of the underlying corpus (the Web).

In the case of the GOV database, we were faced to the fact that some words appearing in the initial list of examples do not appear within the GOV corpus or appears with a very low frequency (due to the fact that this is a government proceedings corpus where, for instance "mice" does not appear and we cannot deal with "mouse is to mice as woman is to women"). So we are forced to reduce the size of our testing set to 70 (28 positive and 42 negatives examples) and we get when using I_1 for analogy definition:

I_1:		positive examples	negative examples
	+	24	24
	−	4	18

This leads to an accuracy rate of 42/70 * 100 % = 60%. With I_2, the confusion matrix becomes:

I_2:		positive examples	negative examples
	+	24	19
	−	4	23

Finally, we obtain an accuracy rate of 47/70 *100 $\simeq$ 70%, without any other preliminary tests. This is not better than the results in [7] with the original examples but it is obvious that our testing set is a bit small. But let us investigate what the expected behaviour of our formula is.

- Let us start with our definition I_1. From an implementation viewpoint, we expect the sum $K(a/b) - K(c/d) + K(b/a) - K(d/c)$ to be close to 0 to classify $a : b :: c : d$ as an analogical proportion. This lead us to compute:

$$log_2 p(a) + log_2 p(b) - log_2 p(c) - log_2 p(d) - 2log_2 p(a,b) + 2log_2 p(c,d) =$$

$$log_2 \frac{p(a)p(b)p(c,d)^2}{p(c)p(d)p(a,b)^2}$$

  This number is close to 0 when

$$N_1 = \frac{p(a)p(b)p(c,d)^2}{p(c)p(d)p(a,b)^2} \text{ is close to 1.}$$

  Among the numerous options to get this result in our context, one is to have both $p(a)$ and $p(b)$ close together to $p(a)p(b)$, both $p(c)$ and $p(d)$ close to $p(c)p(d)$. In terms of probability, this means a and b are not independent. Indeed $p(a)$ close to $p(a,b)$ implies 2 facts:
  1) $K(b/a)$, estimated via $-log_2 p(b/a)$ is close to 0 i.e. it is easy to get b from a.
  2) the probability of a page containing a but not b is close to 0.
  In our context, the probability of a page containing *snow* without containing *flake* (and vice versa) is quite low. That is why proportion like "drop is to rain as flake is to snow" are easily classified as analogy by our system.
- Let us carry on with definition I_2, where a new condition is added to I_1 to enforce the central permutation property. This new condition is similar to I_1 where we have permuted b and c. So a computation as above leads to consider the number

$$N_2 = \frac{p(a)p(c,d)p(b,d)}{p(d)p(a,b)p(a,c)} \text{ which has to be close to 1.}$$

  The raw probabilities $p(b)$ and $p(c)$ disappear in the final formula, confirming the fact that they can be permuted in the proportion, which was not the case for the previous number N_1.

5.3 A new option

The above analysis leads us to consider a new definition for analogical proportion. Our first definition I_1 using Kolmogorov complexity was not sufficient to insure

the central permutation property, expected from an analogical proportion. That is why we have added to I_1 (whose implementation is via the test $N_1 \backsimeq 1$) a second condition to enforce central permutation, getting I_2 (whose implementation is via the test $N_2 \backsimeq 1$). But, as seen in section 2.3, there is a another property related to the negation operator which is $a : b :: c : d \rightarrow c : \neg b :: d : \neg a$. Obviously, this property has no translation in a Kolmogorov framework. Nevertheless, in terms of frequencies, it makes sense to consider the frequency of the pages containing c among the pages which do not contain b, leading to an estimation of $p(c/\neg b)$. From a practical viewpoint, the term $\neg b$ refers to the pages which do not contain any occurrence of b. Instead of taking into account the presence of a word c in the context where a word b appears (i.e. $p(c/b)$), we consider the presence of the word c in the context where a word b does not appear i.e. $p(c/\neg b)$. Obviously this gives an information about the existing link between c and b, which could be helpful in testing analogical proportion involving c and b. That is why we enforce the property above with an extended definition for analogical proportion:

$$(I_2) \wedge [(p(c/\neg b) = p(d/\neg a)) \wedge (p(b/\neg c) = p(a/\neg d)] \ (I_3)$$

When using I_3 on the same test set, the confusion matrix becomes:

I_3:		positive examples	negative examples
	$+$	24	12
	$-$	4	30

Finally, we obtain an accuracy rate of $54/70 * 100 \simeq 77\%$ which is acceptable. At this stage, we think a larger test set is necessary to go for more accurate definitions.

6 Related works

To model analogical proportions, a large panel of algebraic models have been provided, from semi-groups to lattices, through words over finite alphabets and finite trees. One can for instance refer to [14, 3, 15, 16]. Moreover, in [17], formal proportions of structured objects have been investigated, going through second order logic substitutions: this approach allows its authors to capture high level mapping between highly structured universes. [3] might be viewed as a particular case of it and a practical approach to build up an analogical-proportion based learning engine. Nevertheless, apart the previously cited [2], these works do no really provide a framework to deal with natural language analogies. The use of analogy in cognition and especially in learning has been rigorously identified and widely discussed, e.g. [18]. But, as far as we know, [19] was the first to establish a link between analogical reasoning and Kolmogorov complexity, leaving the rather strict logical framework coming from [20] or, more recently, [5]. The author advocates a kind of "simplicity principle" to serve as a starting point for modeling analogy via Kolmogorov theory considered as a mathematical formulation of the Occam's razor. This approach is in complete line with the works of [21] in which "choose the pattern that provides the briefest representation of the available information" is acknowledged as the rational

basis of a wide range of cognitive processes, including analogical reasoning. On the other hand, there are a number of works in diverse fields (linguistics, cognitive sciences) using word frequencies to design various similarity/dissimilarity measures [22, 23]. However, despite some obvious relations, our Kolmogorov complexity-based approach is quite different. Moreover, it has been shown in [7], that a definition for analogy based on Jaccard index provides less satisfactory results in terms of analogies classification. This suggests that the information-theoretic model coming from Kolmogorov theory is an appropriate definition not only for strings, but also for concepts represented as words. Still a careful comparison with the approach in [23] which has been especially developed for analogical proportions would be in order.

7 Conclusion

In this paper, we have established a parallel between a purely abstract view of analogical proportion, suitable for structured domains, and a more practical one, dedicated to deal with natural language analogies. Based on Kolmogorov complexity as a concise definition for the notion of "information content", we provide diverse formulas suitable to define an analogical proportion in natural language. Using the relationship between Kolmogorov complexity and the universal distribution, we get a method to estimate this complexity and then a way to implement a practical tool to check our definitions. Using frequency as an approximation of a probability, our definitions are implemented via the computation of diverse numbers, using a collection of texts. A careful analysis of our implementation formulas, leads to take into account another kind of information, which cannot be described via Kolmogorov theory, but which makes sense in practice. This last definition leads to results better than those previously obtained with the Web as target corpus. Obviously, it remains to check our ideas on a larger scale, at least in two directions:

- on the formal side and in order to get a more accurate classifier, to transfer in our definition other logical properties, expected from an analogical proportion, but not expressible within Kolmogorov framework (like $p(a/\neg b) = p(c/\neg d)$ and $p(b/\neg a) = p(d/\neg c)$).
- on the practical side, to work with other structured databases, more general than the TREC GOV and having a larger diversity of words, allowing to investigate a larger set of examples.

When properly developped, our definitions could provide the ability to automatically discover analogies between concepts, starting from natural language raw text without the need of pre-processing the corpus.

Acknowledgements: The authors are indebted to Mohand Boughanem and Cécile Laffaire for providing them with the opportunity to access the .GOV corpus.

References

1. Goel, A.K.: Design, analogy and creativity. IEEE Expert **12** (1997) 62–70
2. Sowa, J.F., Majumdar, A.K.: Analogical reasoning. In: Proc. Inter. Conf. on Conceptual Structures. LNAI 2746, Dresden, Springer-Verlag (2003) 16–36
3. Stroppa, N., Yvon, F.: Analogical learning and formal proportions: Definitions and methodological issues. ENST Paris report (2005)
4. Prade, H., Richard, G.: Analogical proportions: another logical view. In Bramer, M., Ellis, R., Petridis, M., eds.: Res. and Dev. in Intelligent Systems XXVI, Proc. 29th Ann. Inter. Conf. on AI (SGAI'09), Cambridge, UK, December 2009, Springer (2010) 121–134
5. Prade, H., Richard, G.: Reasoning with logical proportions. In: Proc. Inter. Conf. on Principles of Knowledge Representation and Reasoning. (KR'10), Toronto, Canada (2010) 546–555
6. Hofstadter, D., Mitchell, M.: The Copycat project: A model of mental fluidity and analogy-making. In Hofstadter, D., The Fluid Analogies Research Group, eds.: Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought, New York, NY, Basic Books, Inc. (1995) 205–267
7. Prade, H., Richard, G.: Testing analogical proportions with Google using Kolmogorov information theory. In: Proc. of Int. Conf. FLAIRS22, Fort Myers, USA, AAAI Press (2009) 272–277
8. Miclet, L., Prade, H.: Handling analogical proportions in classical logic and fuzzy logics settings. In: Proc. 10th ECSQARU, Verona. Volume LNCS 5590., Springer (2009) 638–650
9. Solomonoff, R.J.: A formal theory of inductive inference, part l. I and II; Information and Control **7** (1964) 1–22
10. Kolmogorov, A.N.: Three approaches to the quantitative definition of information. Problems in Information Transmission **1**(1) (1965) 1–7
11. Li, M., Vitányi, P.: Introduction to Kolmogorov Complexity and Its Applications. ISBN 0-387-94053-7. Springer-Verlag (1997)
12. Bennett, C., Gacs, P., Li, M., Vitányi, P., Zurek, W.: Information distance. IEEE Transactions on Information Theory **44**(4) (1998) 1407–1423
13. Cilibrasi, R., Vitanyi, P.: Automatic meaning discovery using google. In: Manuscript, CWI, 2004; http://arxiv.org/abs/cs.CL/0412098. (2004)
14. Stroppa, N., Yvon, F.: An analogical learner for morphological analysis. In: Proc. 9th Conf. Comput. Natural Language Learning (CoNLL-2005). (2005) 120–127
15. Miclet, L., Delhay, A.: Relation d'analogie et distance sur un alphabet defini par des traits. Technical Report 1632, IRISA (July 2004)
16. Barbot, N., Miclet, L.: La proportion analogique dans les groupes: applications aux permutations et aux matrices. Technical Report 1914, IRISA (July 2009)
17. Schmid, U., Gust, H., Kühnberger, K., Burghardt, J.: An algebraic framework for solving proportional and predictive analogies. Eur. Conf. Cogn. Sci. 295-300 (2003)
18. Gentner, D., Holyoak, K.J., Kokinov, B.: (Eds.) The Analogical Mind: Perspectives from Cognitive Sciences. MIT Press (2001)
19. Cornuéjols, A.: Analogie, principe d'économie et complexité algorithmique. In: In Actes des 11èmes Journées Francaises de l'Apprentissage. Sète, France. (1996)
20. Davies, T.R., Russell, S.J.: A logical approach to reasoning by analogy. In: IJCAI-87, Morgan Kaufmann (1987) 264–270
21. Chater, N.: The search for simplicity : A fundamental cognitive principle? In Taylor, Francis, H., eds.: The Quarterly journal of experimental psychology. Volume 52, n2. (1999) 273–302
22. Terra, E., Clarke, C.L.A.: Frequency estimates for statistical word similarity measures. In: Proceedings of the Human Language Technology and North American Chapter of Association of Computational Linguistics Conference HLT/NAACL. (2003) 244–251
23. Turney, P.: Similarity of semantic relations. Computational Linguistics **32**(3) (2006) 379–416

Evolving Temporal Association Rules with Genetic Algorithms

Stephen G. Matthews, Mario A. Gongora and Adrian A. Hopgood

Abstract A novel framework for mining temporal association rules by discovering itemsets with a genetic algorithm is introduced. Metaheuristics have been applied to association rule mining, we show the efficacy of extending this to another variant - temporal association rule mining. Our framework is an enhancement to existing temporal association rule mining methods as it employs a genetic algorithm to simultaneously search the rule space and temporal space. A methodology for validating the ability of the proposed framework isolates target temporal itemsets in synthetic datasets. The Iterative Rule Learning method successfully discovers these targets in datasets with varying levels of difficulty.

1 Introduction

Data Mining is the process of obtaining high level knowledge by automatically discovering information from data in the form of rules and patterns. Data mining seeks to discover knowledge that is accurate, comprehensible and interesting [9]. Association rule mining is a well established method of data mining that identifies significant correlations between items in transactional data [1]. An example of this is a rule that states "customers who purchase bread and milk also purchase cheese". The use of such rules provides insight into transaction data to allow businesses to make better informed decisions. The usefulness of association rule mining extends to many areas ranging from biomedical and environmental to social networking and retail. With an increasing volume of information this is prevalent in time series data as well as static problems.

However, classical association rule mining assumes the dataset to be static where discovered rules are relevant across the entire dataset. In many cases this does not

Stephen G. Matthews, Mario A. Gongora and Adrian A. Hopgood
Centre for Computational Intelligence, De Montfort University, Leicester, UK e-mail: sgm@dmu.ac.uk; mgongora@dmu.ac.uk; aah@dmu.ac.uk

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_8,

reflect real-world data. Often there can be a temporal pattern behind the occurrence of association rules. The scope is far reaching, many systems producing time series data have underlying processes/events that are dynamic. For example, association rules may occur more frequently in the days leading to a large sports event, or when an unforeseen event occurs, such as network intrusions. Discovering and adapting to changes with well-informed information is important in many domains, and within business it is critical for success. Association rules that incorporate temporal information have greater descriptive and inferential power [17] and can offer an additional element of interestingness.

Existing approaches to mining temporal association rules rely on identifying all itemsets that frequently occur throughout the dataset. With a large number of attributes this is computationally expensive and can lead to combinatorial explosion with classical methods. In this paper, we present an approach that incorporates a genetic algorithm [14] to mine frequent itemsets for temporal association rules without exhaustively searching the itemset space and temporal space. The temporal rules sought from the itemsets are those that occur more frequently over an interval of the dataset. They are seen as an area of greater itemset density. This research seeks to analyse the efficacy of a genetic algorithm for mining temporal association rules, where there has been very little research into this aspect of association rule mining, e.g. [5]. This is a challenging problem for a genetic algorithm because it involves searching the itemset space as well as the temporal space. Searching additional spaces other than the rule space has been shown in other variants of association rule mining such as quantitative [19, 3] to be effective. Our approach offers benefits where temporal patterns are of interest. It can be scaled up to larger problems and can include evolving additional parameters.

This paper is organised as follows. An overview of related work covering association rule mining and evolutionary computing methods, such as genetic algorithms, is discussed in Section 2. In Section 3 the genetic-algorithm based approach for mining temporal association rules is presented. An experiment to analyse the efficacy is presented and discussed with results in Section 4 and we conclude our work in Section 5.

2 Related Work

2.1 Temporal Association Rule Mining

A synopsis of preliminary work on classical association rule mining is discussed before developing the concept further to include a temporal aspect. Association rule mining is an exploratory and descriptive rule induction process of identifying significant correlations between items in boolean transaction datasets [1] used for data analysis and interpretation.

The formal definitions of association rule mining are presented as required preliminary knowledge. We assume a set of items $I = \{i_1, i_2, ..., i_M\}$ for all product items in market basket data and a set of all transactions $D = \{d_1, d_2, ..., d_N\}$. Each transaction, d_i, comprises a subset of items, referred to as an itemset, from I representing a collection of items found in a customer's shopping basket. Association rules are expressed as an implication of the form $X \Rightarrow Y$ where the consequent and antecedent are sets of boolean items where $X \cap Y = \emptyset$. For example, the rule $\{\text{bread, milk}\} \Rightarrow \{\text{cheese}\}$ implies that when bread and milk are purchased cheese is also purchased. This suggests there is a strong relationship between the sale of bread and milk, and the sale of cheese. This is the most simple form of association rules.

To extract such rules from datasets the support-confidence framework was introduced with the Apriori algorithm in [2]. Support determines the strength of a relationship by measuring how often the rule occurs in a dataset. Confidence determines how frequently the items in the consequent occur in transactions containing the antecedent, which measures the reliability of the inference. Support and confidence are defined in Equations 1 and 2 respectively.

$$\text{Support}, s(X \Rightarrow Y) = \frac{\sigma(X \cup Y)}{N} \tag{1}$$

$$\text{Confidence}, c(X \Rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \tag{2}$$

Minimum support and minimum confidence are introduced to limit the itemsets and rules produced to those that are significant and interesting. In the general case, a rule that occurs once in the dataset would not be considered interesting. Based on this support-confidence framework, rules are extracted by the following stages. First, the frequent itemsets are generated that have a support above the minimum support. The rules are then identified from the frequent itemsets that satisfy the minimum confidence constraint. This is a common approach for rule induction that employs a level-wise, breadth-first strategy but there are other methods that use equivalence classes, depth-first and compression techniques [21].

Association rules that have an underlying temporal behaviour can be expressed with various temporal patterns and frameworks. A key issue of classical methods that are based on the support-confidence framework is that temporal patterns having low support values can escape through the minimum support threshold. For example, consider a rule that has high support in the month of December but for the remainder of the year it is relatively much lower. This rule may not be discovered with classical association rule mining algorithms when there are rules that persistently occur throughout the dataset and consequently have higher support. Assuming that the minimum support is sufficiently low for the rule in December to be discovered, further analysis is required to ascertain any temporal property. One such property, *lifespan*, was introduced in [4] as an extension on the Apriori algorithm [2]. This is a measure of support that is relative to the lifespan

of the itemset defined by a time interval, known as temporal support. So for the example rule occurring in December, its temporal support would be relevant for a time interval representing December.

A similar problem can be found where individual items are not present throughout the entire dataset. For example, an item may be available for sale in a supermarket only during a particular seasonal period, such as British asparagus during summer. In [6, 15] the temporal element of individual items is considered rather than that of the itemsets. This variation of the problem shares the same issue with support as seen in [4] where low-support itemsets can be lost under the minimum support threshold. A related area that also focuses on the analysis of support values within a temporal framework is that of [8]. Their work introduces the concept of emerging patterns which describe itemsets as those where support increases significantly from one dataset to another. New trends can be identified by itemsets that are starting to appear more frequently.

Other methods seek to identify temporal patterns with techniques that do not directly analyse support. In [20] cyclic association rules are defined as rules that occur in user-defined intervals at regular periods throughout a dataset. For example, "*at weekends,* customers who purchase bacon and eggs also purchase sausages". This is achieved by mining association rules from partitions of the dataset and then pattern matching the rules from each partition. These are fully periodic rules because they repeatedly occur at regular intervals. Partially periodic rules [12] relax the regularity found in fully periodic so the cyclic behaviour is found in only some segments of the dataset and is not always repeated regularly. Defining the temporal intervals with calendar-based schemas is less restrictive and reduces the requirement of prior knowledge [18].

2.2 Association Rule Mining with Evolutionary Computation

Evolutionary computation is a subfield of computational intelligence that uses evolutionary systems as computational processes for solving complex problems [7]. There has been recent interest in the use of evolutionary algorithms, as well as swarm intelligence techniques, for association rule mining [19, 22, 16]. Evolutionary algorithms are metaheuristic methods that are suitable for association rule mining because they can search complex spaces and they address difficult optimisation problems. We discuss applications of evolutionary computation to association rule mining to highlight its suitability for this novel application.

With an evolutionary approach, there are generally two main strategies to encoding rules in a solution [13] that determine how rules are evolved. The Pittsburgh approach represents each individual as a ruleset and the Michigan approach encodes each individual as a single chromosome. Iterative Rule Learning is based on the Michigan approach except that the best solution is chosen after multiple runs of the genetic algorithm. The genetic cooperative-competitive approach encodes the rules in the population which collectively forms the solution.

Considerable research has focused on the use of evolutionary algorithms for mining quantitative association rules since these are present in many real-world applications. These are different from boolean association rules because they include a quantitative value describing the amount of each item. A method of mining quantitative data requires the values to be discretised into meaningful representations that are comprehensible, but this is a difficult task as the number of attributes and their parameters may not be known. Evolutionary algorithms have been shown to be capable of defining the intervals for quantitative attributes whilst simultaneously extracting association rules [19]. A recent study [3] has demonstrated the effectiveness of several genetic algorithms based on the Michigan and Iterative Rule Learning approaches for mining association rules and itemsets compared with classical mining algorithms like Apriori.

Minimum support is a very influential factor affecting the results of the mining process and it is challenging to specify a priori. Support values that are too low will yield many rules, but support values that are too high will produce too few, if any. In [22] a genetic algorithm is employed where fitness is determined based on relative confidence of association rules across the entire dataset, thus no minimum support is specified. This is shown to be suitable for both boolean and quantitative association rules. Particle swarm optimisation is an alternative method that also achieves similar results for the same purpose of not defining minimum support [16].

As well as mining quantitative data and removing the need for specifying minimum support, evolutionary computation has seen applications in only a few temporal association rule mining tasks. In [5] the Pittsburgh approach is used to mine association rules from partitions of a dataset. The resulting rules are then analysed to discover changes between partitions, similar to those of [20]. Higher-level rules are then produced from the changes in association rules to describe the underlying temporal patterns. The changes in association rules have also been evolved for the purposes of trading on the financial markets [10].

These approaches demonstrate the ability of evolutionary computation in searching for association rules and/or optimising parameters of rules (membership functions), or the induction process (support values). Our novel approach draws on the strengths of evolutionary algorithms for mining association rules that is evident from recent research. The next section describes the evolutionary algorithm approach we have adopted for mining temporal association rules.

3 Evolving Temporal Association Rules

We propose the use of a genetic algorithm to evolve temporal association rules that have high relative support over a time interval. A genetic algorithm is chosen because it is a promising solution for global search and it is capable of discovering itemsets with corresponding parameters as seen in [19, 3]. Our approach searches for itemsets occurring more frequently in an exhibition period by optimising (maximising) the relative support over a discovered time interval.

This has similarities with evolutionary methods for mining attribute intervals since we are evolving a temporal element that is also interval based.

Several test runs of the genetic algorithm were used to determine the configuration of parameters. The number of iterations of the genetic algorithm is set to 15. The genetic algorithm's population is set at 500 individuals and it is terminated at 200 generations. Elitism accounts for 1% of the new population, copy produces 25%, crossover produces 45% and mutation produces 39%. Descriptions of the genetic algorithm's configuration are now presented.

Iterative Rule Learning The Iterative Rule Learning approach is used where each chromosome represents a single itemset and the best solution from numerous runs of the genetic algorithm is selected. This approach relies on the stochastic process of genetic algorithms to yield different solutions. An advantage of Iterative Rule Learning is it produces a reduced rule set, depending on the number of iterations, that contains rules of significant temporal interest and that are easily comprehensible. In the case of classical association rule mining, with no temporal element, Iterative Rule Learning would aim to evolve a reduced rule set containing the most frequent itemsets. We do not penalise the fitness of solutions that have evolved in previous runs and so permit the same solution to be evolved. Doing so gives a clear indication of the efficacy of evolving a single isolated temporal pattern that we use as a specific target. The methodology for isolating the target is explained in more detail in Section 4.1.

Chromosome Figure 1 shows the configuration of genes in the chromosome. An integer representation encodes each item, x_i, where the ordering of items is unimportant. Lower and upper endpoints, t_0 and t_1 respectively, define the edges of the interval in which the itemset occurs most frequently. The chromosome length is fixed allowing only a specified itemset length to evolve on each run. The itemsets are evolved first because the measure (temporal support) used for identifying patterns evaluates itemsets only. The association rules are then generated from the itemsets by calculating the confidence measure after the genetic algorithm has executed.

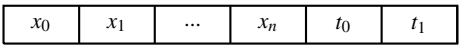


Fig. 1 Chromosome

Population Initialisation The initial population is randomly generated using the Mersenne Twister pseudorandom number generator. Setting different random seeds for each run ensures the experiment is repeatable. Upon randomly generating an item in a chromosome, it is checked against other items already generated in the same chromosome and if the item is present a new number is randomly generated until it is unique. This is repeated for each item in the chromosome to ensure all items are unique. The number of items in the dataset (e.g. inventory) must be greater than the itemset size otherwise this will result in chromosomes where

the only difference is the ordering of items. The lower and upper endpoints are randomly generated using the same method of repeating the number generation until the solution is feasible. The constraint on the endpoints is the minimum temporal support in Equation 3, this is discussed further with the fitness evaluation.

$$t_1 - t_0 >= min\_temp\_sup \tag{3}$$

Fitness Evaluation Fitness is evaluated using the relative support of the itemset over its lifespan. Equation 4 is the temporal support metric defined in [4].

$$s(X, l_X) = \frac{\sigma(X)}{l_X} \tag{4}$$

We introduce l as a time interval i.e. $l_X = [t_0, t_1]$ where t_0 is the lower endpoint and t_1 is the upper endpoint. The genetic algorithm maximises temporal support. A minimum temporal support [4] is used to prevent evolving solutions to a minimal lifespan that only cover one transaction. For example, a lifespan of 1 covers a single transaction, this produces a support of 100% for any itemset i.e. maximum fitness.

Selection Fitness proportionate selection is used to select individuals from a population for copying across to a new population or applying genetic operators. A method based on roulette wheel selection is employed. A random float value is generated between 0 and the sum of all fitness values. The fitness values are then accumulated until the accumulation is greater than the random float value. The individual selected is that which pushes the accumulation above the random number.

Genetic Operators Elitism is used to automatically copy over the best individuals from the current population to the next population without selection. A percentage of individuals are also selected and copied into the next generation.

Uniform crossover is adapted to ensure that only feasible solutions are produced, i.e. combinations of integers without duplicates. The method for crossing over only the itemsets is presented in Algorithm 1 and the stages are now briefly described. The advantage of this method is that the ordering of items remains unless a duplicate is present in the itemset.

Stage 1 (lines1 - 4) Merge the chromosomes from two selected parents into an intermediate array so that no two items from the same parent are adjacent.

Stage 2 (lines 5 - 11) Check each item in the array for duplicate values against the remaining items. If a duplicate is found the duplicate item is swapped with the next item. The result is that all duplicate items are now adjacent and the items can now be selected from the intermediate array to form an offspring.

Stage 3 (lines 12 - 18) Select items from the intermediate array by iterating over every even index value. A random integer from $[0, 1]$ is added to the index and the indexed item is added to the offspring. If a 0 is generated, it is checked for duplicates with the preceding item and if a duplicate is found it adds 1 to the index otherwise it adds 0.

A random integer from $[0,1]$ determines whether the genes representing the lower and upper endpoints are copied from a single parent or they are crossed over from two parents. If they are crossed over then the feasibility of offspring is ensured by satisfying the constraint in Equation 3.

Algorithm 1 Algorithm for performing crossover on itemsets

Require: $Parent1.length \equiv Parent2.length$
for $i = 0$ to $Parent1.length - 1$ **do**
  $Auxiliary[2i] = Parent1[i]$
  $Auxiliary[2i+1] = Parent2[i]$
end for
for $i = 0$ to $Auxiliary.length - 1$ **do**
  for $j = i+2$ to $Auxiliary.length - 1$ **do**
    if $Auxiliary[i] \equiv Auxiliary[j]$ **then**
      exchange $Auxiliary[j]$ with $Auxiliary[i+1]$
    end if
  end for
end for
for $i = 0$ to $Parent1.length - 1$ **do**
  if $i > 1$ and $Auxiliary[2i-1] \equiv Auxiliary[2i]$ **then**
    $Child[i] = Auxiliary[2i+1]$
  else
    $Child[i] = Auxiliary[2i+$RANDOM(0,1)$]$
  end if
end for

To produce a mutated individual, a chromosome is selected and a randomly chosen gene is replaced with a randomly created value that is feasible. For the genes forming the itemset, the value must be unique and the genes for the endpoints must satisfy Equation 3.

4 Evaluation

To evaluate the efficacy of the proposed approach, several experiments have been conducted on synthetic datasets. The aim is to ascertain whether the algorithm can correctly identify areas where association rules occur more frequently.

4.1 Methodology and Datasets

The IBM Quest Synthetic Data Generator [11] [1] has been used to generate a dataset for experimentation. The generator produces datasets that replicate transactions. This approach was first used in work that focused on a retail environment [2]. A synthetic dataset is chosen rather than a real dataset so that a controlled experiment can be conducted to validate the efficacy of our approach. Individual temporal itemsets that exhibit relatively high support over an exhibition period are isolated and used as target solutions.

A dataset has been produced with the following features: 1000 transactions, 50 items, an average size of transactions of 10 and a maximal pattern length of 4. A maximal pattern cannot be part of any rule of greater length; it has no supersets that are frequent. There is no guarantee that the generated dataset contains any temporal patterns so, to include temporal information, two datasets have been augmented from the original dataset by the following process:

1. Run Apriori algorithm on dataset to produce frequent itemsets.
2. Select a frequent itemset with desired level of support.
3. Insert the itemset as a transaction near to the centre of the dataset. Transactions are constructed exclusively from the entire frequent itemset with no additional items so no unexpected correlations between items are introduced.

The itemsets with maximum support (6.8%) and midrange support (3.4%) were selected as varying levels of difficulty for the experiment. Itemsets were inserted into the dataset within bin sizes of 50 so that the lifespan of an itemset is of sufficient size for identifying temporal association rules. Figure 2 shows a histogram of the original dataset compared with augmented dataset containing the itemset {12, 21, 25, 45} with maximum support. The horizontal axis shows the number of occurrences in bin sizes of 50. This bin size . This shows the increased occurrence of the itemset, the isolated target, that is to be discovered with the genetic algorithm. Figure 3 shows the original dataset against the other augmented dataset containing itemset {8, 12, 39, 45} with midrange support. The peaks in these figures illustrate the more frequent occurrence of itemsets over a relatively small period of time that are target itemsets and intervals.

The itemset with midrange support (3.4%) is chosen because it is expected that this will be a more difficult dataset for the genetic algorithm. The genetic algorithm is more likely to follow local searches of itemsets because it is likely they have higher relative support values over the same lengths of time intervals. The support measure is used to evaluate fitness because this is the metric used to augment the dataset with significant temporal patterns.

[1] This is the data generator pioneered in [2] but the original link ceases to exist (http://www.almaden.ibm.com/cs/quest/syndata.html)

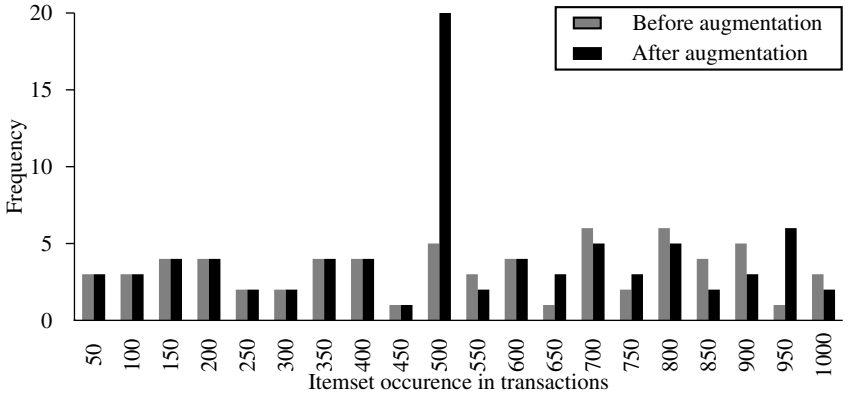


Fig. 2 Histogram of itemset {12, 21, 25, 45} with high support

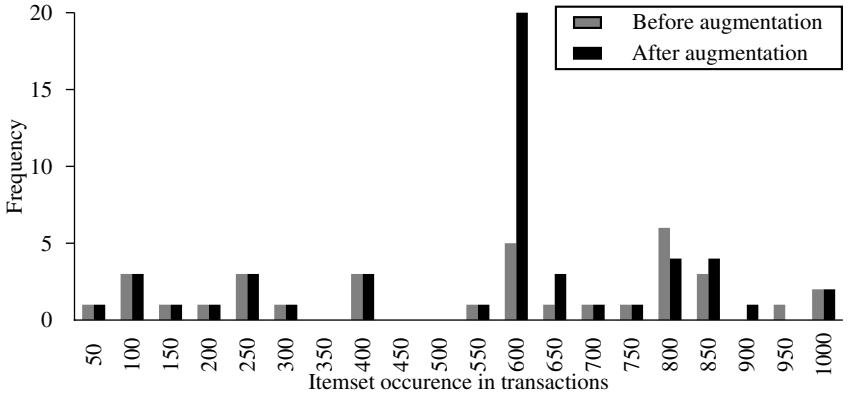


Fig. 3 Histogram of itemset {8, 12, 39, 45} with midrange support

4.2 Results

The genetic algorithm was executed 15 times with different random seeds on both augmented datasets for a maximum of 200 generations. Itemsets of length 4 were mined because this is the average maximal frequent itemset defined in the parameters of the dataset generator. The minimum temporal support was chosen based on the bin sizes used in the method for augmenting the datasets, this was set to 50. Table 1 shows the evolved itemset from each run with its corresponding interval and support values for the dataset augmented with the high support itemset. The results for this dataset show the genetic algorithm is able to consistently evolve the itemset and the endpoints for the inserted itemset in the majority of runs. The suboptimal solutions have much lower temporal support than the inserted high support itemsets. Although the termination criteria was set to 200 generations the best individuals were evolved in far fewer generations.

Seed	Itemset	Lower endpoint	Upper endpoint	Temporal Support	Generation
0	{12,21,25,45}	449	502	41.5%	51
1	{12,21,38,45}	904	960	14.3%	86
2	{12,21,25,45}	449	502	41.5%	59
3	{12,21,25,45}	449	502	41.5%	72
4	{8,12,21,43}	691	752	16.4%	37
5	{12,21,25,45}	449	502	41.5%	49
6	{12,21,25,45}	449	502	41.5%	35
7	{12,21,25,45}	449	502	41.5%	45
8	{12,21,25,45}	449	502	41.5%	60
9	{12,21,45,48}	449	502	41.5%	67
10	{12,21,38,45}	904	960	14.3%	75
11	{8,12,21,43}	687	738	15.7%	38
12	{8,12,25,45}	233	283	14.0%	20
13	{12,21,25,45}	449	502	41.5%	26
14	{12,21,25,45}	449	502	41.5%	63

Table 1 Genetic algorithm results of dataset inserted with high support itemset {12, 21, 25, 45}

The results of applying the genetic algorithm to the dataset augmented with the midrange support itemset are presented in Table 2. The results show the genetic algorithm is able to evolve the inserted itemset with the corresponding endpoints (seeds 3 and 14). However, this occurs in only a few runs of the genetic algorithm, many fewer than the previous dataset, suggesting it is a more difficult dataset. The support value across the entire dataset in Table 2 shows the genetic algorithm is more likely to evolve temporal patterns that are generally more frequent across the entire dataset. An itemset with high support occurs more frequently and so temporal patterns are found of this itemset. The histogram in Figure 4 shows an example itemset from Table 2 (seeds 4, 5, 7, 12 and 13) with high support and low temporal support (small peak in bin 800) which suggests a local optimum has evolved.

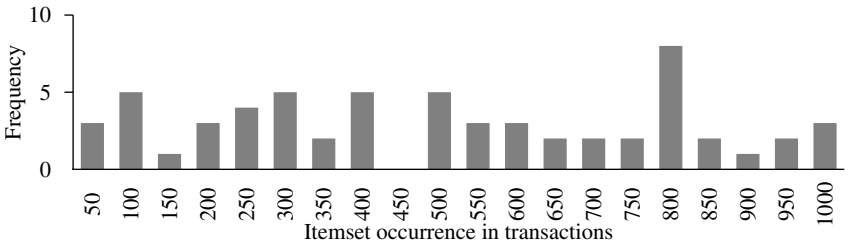


Fig. 4 Histogram of itemset {8, 12, 25, 45} in dataset augmented with midrange support itemset

From the results of executing the genetic algorithm on both datasets we can see the optimal solution is evolved. The repeatability of evolved solutions varies because of the stochastic nature of genetic algorithms but it also varies considerably between

Seed	Itemset	Lower endpoint	Upper endpoint	Temporal Support	Support	Generation
0	{12,21,38,45}	905	961	14.3%	5.7%	38
1	{1,12,21,45}	902	952	14.3%	5.0%	81
2	{8,12,21,43}	750	801	15.7%	6.1%	92
3	{8,12,39,45}	550	601	39.2%	5.1%	85
4	{8,12,25,45}	766	819	17.0%	6.2%	95
5	{8,12,25,45}	766	819	17.0%	6.2%	43
6	{8,12,21,43}	673	735	16.0%	6.1%	39
7	{8,12,25,45}	766	819	17.0%	6.2%	148
8	{8,12,21,43}	673	735	16.1%	6.1%	61
9	{10,12,21,45}	787	838	13.7%	3.5%	26
10	{12,21,38,45}	905	961	14.3%	5.7%	28
11	{8,12,21,43}	692	753	16.4%	6.1%	72
12	{8,12,25,45}	234	284	14.0%	6.2%	76
13	{8,12,25,45}	766	819	17.0%	6.2%	107
14	{8,12,39,45}	533	605	38.5%	5.1%	110

Table 2 Genetic algorithm results of dataset inserted with midrange support itemset {8, 12, 39, 45}

the two datasets. Low support items with high temporal support are more difficult to discover.

5 Conclusion

In this paper we have presented a novel approach to mining temporal association rules by discovering itemsets with a genetic algorithm. The genetic algorithm approach is capable of discovering itemsets that occur more frequently over a short time interval of a transactional dataset. The genetic algorithm method is an enhanced approach for simultaneously searching the itemset space and temporal space. The advantage of this approach is that it does not exhaustively search the dataset or require any prior partitioning.

Having identified this method to be capable, future work will include analysing its effectiveness in terms of quality of rules produced and its scalability through comparative analysis with other methods. We will investigate enhancing the fitness evaluation to reduce the chances of evolving local optima. The Iterative Rule Learning approach is a promising framework for analysing rule quality and, as already seen, individuals can be penalised to avoid searching the same areas of the fitness landscape. Further experiments on varying the number of transactions and items will provide insight into scalability. Our methodology has augmented a single temporal itemset into a synthetic dataset so future plans include using a real dataset to identify meaningful rules.

Acknowledgements This research has been supported by an EPSRC Doctoral Training Account.

References

1. Agrawal, R., Imieliński, T. and Swami, A. (1993) Mining association rules between sets of items in large databases. In: Proceedings of ACM SIGMOD international conference on Management of data, Washington, DC, USA, pp. 206–217.
2. Agrawal, R. and Srikant, R. (1994) Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Data Bases, Santiago, Chile, pp. 487–499.
3. Alcala-Fdez, J., Flugy-Pape, N., Bonarini, A. and Herrera, F. (2010) Analysis of the Effectiveness of the Genetic Algorithms based on Extraction of Association Rules. Fundamenta Informaticae, 98(1), pp. 1–14.
4. Ale, J. and Rossi, G. (2000) An approach to discovering temporal association rules. In: Proceedings of the 2000 ACM Symposium on Applied computing (SAC 00) New York, NY, USA, pp. 294–300.
5. Au, W. and Chan, K. (2002) An evolutionary approach for discovering changing patterns in historical data. In: Proceedings Of The Society Of Photo-Optical Instrumentation Engineers (SPIE), Orlando, FL, USA, pp. 398–409.
6. Chang, C.-Y., Chen, M.-S. and Lee, C.-H. (2002) Mining general temporal association rules for items with different exhibition periods. In: Proceedings of the 2002 IEEE International Conference on Data Mining, Maebashi City, Japan, pp. 59–66.
7. De Jong, K.A. (2006) Evolutionary computation: a unified approach. MIT Press, Cambridge, MA, USA.
8. Dong, G. and Li, J. (1999) Efficient mining of emerging patterns: discovering trends and differences. In: Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, pp. 43–52
9. Freitas, A.A. (2002) Data mining and knowledge discovery with evolutionary algorithms. Springer-Verlag.
10. Ghandar, A., Michalewicz, Z., Schmidt, M., Tô, T.-D. and Zurbrugg, R. (2009) Computational intelligence for evolving trading rules. IEEE Transactions on Evolutionary Computation, 13(1), pp. 71–86.
11. Giannella, C. (2003) IBM Quest Market-Basket Synthetic Data Generator. http://www.cs.nmsu.edu/ cgiannel/assoc_gen.html. Cited 29 May 2009
12. Han, J., Gong, W. and Yin, Y. (1998) Mining segment-wise periodic patterns in time-related databases. In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, New York, NY, USA. pp. 214–218.
13. Herrera, F. (2008) Genetic fuzzy systems: taxonomy, current research trends and prospects. Evolutionary Intelligence, 1(1), pp. 27–46.
14. Holland, J.H. (1975) Adaptation in natural and articial systems. University of Michigan Press, Ann Arbor.
15. Huang, J.-W., Dai, B.-R. and Chen, M.-S. (2007) Twain: Two-end association miner with precise frequent exhibition periods. ACM Transactions on Knowledge Discovery from Data, 1(2), Article 8.
16. Kuo, R., Chao, C. and Chiu, Y. (2009) Application of particle swarm optimization to association rule mining. Applied Soft Computing, In Press, Corrected Proof.
17. Laxman, S. and Sastry, P.S. (2006) A survey of temporal data mining. Sādhanā, 31, pp. 173–198.
18. Li, Y., Ning, P., Wang, X. S. and Jajodia, S. (2003) Discovering calendar-based temporal association rules. Data & Knowledge Engineering, 44(2), pp. 193–218.
19. Mata, J., Alvarez, J. L. and Riquelme, J. C. (2002) An evolutionary algorithm to discover numeric association rules. In: Proceedings of the 2002 ACM Symposium on Applied Computing, New York, NY, USA, pp. 590–594.
20. Özden, B., Ramaswamy, S. and Silberschatz, A. (1998) Cyclic Association Rules. In: Proceedings of the Fourteenth International Conference on Data Engineering, Washington, DC, USA, pp. 412–421.

21. Tan, P.-N., Steinbach, M. and Kumar, V. (2005) Introduction to Data Mining, Addison Wesley, Boston, MA, USA.
22. Yan, X., Zhang, C. and Zhang, S. (2009) Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support. Expert Systems with Applications, 36(2), pp. 3066–3076.

PLANNING AND SCHEDULING

PIPSS*: A System Based on Temporal Estimates

Yolanda E-Martín, María D. R-Moreno, and Bonifacio Castaño

Abstract AI planning and scheduling are two closely related areas. Planning provides a set of actions that achieves a set of goals, and scheduling assigns time and resources to the actions. Currently, most of the real world problems require the use of shared and limited resources with time constraints when planning. Then, systems that can integrate planning and scheduling techniques to deal with this kind of problems are needed.

This paper describes the extension performed in PIPSS (Parallel Integration Planning and Scheduling System) called PIPSS*. PIPSS combines traditional state space heuristic search planning with constraint satisfaction algorithms. The extension is based on heuristic functions that allows the planner to reduce the search space based on time estimations that imposes temporal constraints to the scheduler. The purpose is to achieve a tighter integration respect to the previous version and minimize the makespan. Results show that PIPSS* outperforms state of the art planners under the temporal satisficing track in the IPC-08 competition for the tested domains.

1 Introduction

Some planning systems use heuristic functions to search in the state space. These planners are called Heuristic Search Planners (HSPs). HSPs are based on the use of evaluation or heuristic functions, combined with search algorithms to explore the search space toward a desired state.

Yolanda E-Martín
Departamento de Automática, Universidad de Alcalá e-mail: yolanda@aut.uah.es

María D. R-Moreno
Departamento de Automática, Universidad de Alcalá e-mail: mdolores@aut.uah.es

Bonifacio Castaño
Departamento de Matemáticas, Universidad de Alcalá e-mail: bonifacio.castano@uah.es

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_9,

This work is focus on the PIPSS [5] system. PIPSS is composed of HPP [1] and OOSCAR [4] as planner and scheduler respectively. The relaxation heuristic used on HPP is based on ignoring delete actions. This technique defines a relaxation problem ignoring the action delete effects. The heuristic cost is estimated as the length, in actions number, between a certain state s and any goal. The estimation is extracted from a planning graph.

The objective of PIPSS* is to extract temporal estimations from the planning graph, moreover the cost heuristic, that allow us to reduce the makespan of a problem. Thus, from calculated estimates, we will force the scheduler to find a plan whose makespan does not exceed the calculated estimation in the planner. Thereby, if the scheduler does not find a plan that satisfies the restriction, shall inform the planner in order to find another plan that does comply.

That is, the underlying idea developed on top of PIPSS is to get plans for achieving the goals in as few steps as possible.

The paper is structured as follows. The next section details the PIPSS system components. Then, the PIPSS extension (PIPSS*) is described. Next, experimental results are discussed. Finally, some conclusions are presented.

2 PIPSS

PIPSS (Parallel Integrated Planning and Scheduling System) is a system that integrates planning and scheduling. It is able to solve planning problems with time and multicapacity resources information through scheduling techniques.

PIPSS emerges from the union between the HPP planner and the main scheduling algorithm from the OOSCAR system. The next subsections explain each of its components.

2.1 HPP: Heuristic Progressive Planner

HPP is a heuristic progressive PDDL planner based on FF [13]. But HPP introduces changes in the operators instantiation. It includes a new module called *analysis reachability module* that is able to exclude irrelevant domain-dependent operators in the planning process. The analysis performed by this module avoids the expansion of several parts of the search tree[1] and can obtain better results in the planning process.

In particular, HPP builds before the planning process, three sets of operators called *A*, *B* and *C vectors*. The *A vector* contains all the possible operators of the instantiation process (as FF does). The *B vector* has fewer operators than *A vector*, which are the result of employing the relaxed GraphPlan [2] that FF uses when it

[1] A Search tree is defined as a graph that considers all possible paths in the network. The tree nodes represent states, and its branches executed actions that achieve states.

calculates its heuristic. Finally, the *C vector* is generated using an additive heuristic h_{add} as the one used in HSP planner [3] for computing the heuristic cost.

2.2 ISES Algorithm

OOSCAR (Object-Oriented SCheduling ARchitecture) is a scheduling system that works with time and resources to deal with the RCPSP/max problem (Resource Constrained Project Scheduling Problem). The RCPSP problem is a kind of scheduling problem that attempts to look for a way of ordering activities along time, which are restricted by precedence relations (some of them cannot start until some others have finished) and which use renewable resources (resources whose availability is a fixed quantity for every unit of time). The goal is to find initial times for activities so that the makespan is minimized.

The main algorithm used by OOSCAR to find feasible solutions for RCPSP/max problems is ISES (Iterative Sampling Earliest Solutions) [6]. Basically, ISES is a sample optimization algorithm that iterates another algorithm called ESA (Earliest Start Algorithm), which is in charge of returning time- and resource- consistent solutions. ESA solves temporal restrictions using a temporal network (TN)[2] and avoids conflicts due to resources by imposing additional precedence relations between pairs of activities that are responsible of such conflicts. ISES just asks ESA for several solutions to try to find one with a better makespan.

2.3 PIPSS Architecture

The outstanding features of PIPSS, which allows to carry out the extension described in the next section, is its open and object-oriented architecture for planning and scheduling integration . This architecture is based on interfaces that allow interoperability of different planning algorithms, scheduling techniques and search or integration schemes of both. Also this gives the ability to be configured to run on any combination of the possibilities available. Figure 1 shows the architecture.

PIPSS has two kind of planning search: Enforced Hill Climbing and Greedy Best-First Search — *Planning Interface*. One type or scheduling, ISES, or the possibility to disable scheduling — *Scheduling Interface*. And also several types of planning and scheduling integrated search schemes in *Search Interface*. Besides this, the system can use any of the three vectors that HPP builds (see previous subsection).

[2] A TN is defined in [12] as a directed graph whose nodes represent Time Points (TPs) and whose arcs represent distance constraint between TPs. Nodes represent points on a time line where temporal changes happen, and arcs represent activities duration and distance constraints between activities and events.

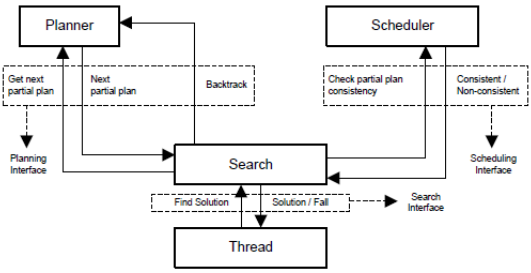


Fig. 1 PIPSS Architecture

3 PIPSS*

In this section we describe the extension introduced in PIPSS, called PIPSS*, that minimizes the makespan of its predecessor. To understand the implementation, it is important to review the relaxed GraphPlan concept.

3.1 Relaxed GraphPlan

GraphPlan [2] is based on a compact structure called a Planning Graph. A Planning Graph is a directed graph composed of two types of nodes distributed on several levels. Starting at level 0, the even levels contain information about the propositions that are reached — *fact layers*, denoted as *Sx*, where *x* is the level number. The odd levels contain actions whose preconditions are present at the previous level — *action layers*, denoted as *Ax*. The edges represent the precondition and effect relations (positive (*add*) and negative (*del*)), between facts and actions.

The Relaxed Graphplan build process has two distinct phases:

- Graph Expansion: builds the planning graph until all goals are reached. In level 0, the fact layer is made up of the initial state (*InitState*) problem facts (in Figure 2 is represented as letter *q*), and the action layer contains all actions applicable to *InitState* (in the example is showed as letters *A* and *B*) . The union of all those actions's add effects (ignoring delete effects) with the facts that are already there, forms the second fact layer (letters *q* and *r* in the figure). This process is repeated until a fact layer is reached that contains all goals (in the example goal state are letters set *q*, *r* and *p*).
- Plan Extraction: a backward-chaining strategy level by level is used. The process consists of: given a set of objectives in t, where t is the last graph level, find a set of actions in level t-1 at reach these goals. Then, the preconditions of these actions form a set of subgoals in t-1. If the new objectives of t-1 can be obtained in t-1 steps, then the original objectives can be achieved in t steps. If on the contrary they cannot be reached, the process looks for a different action combination. The

process continues until a solution is found, or stops if any of actions combinations are valid.

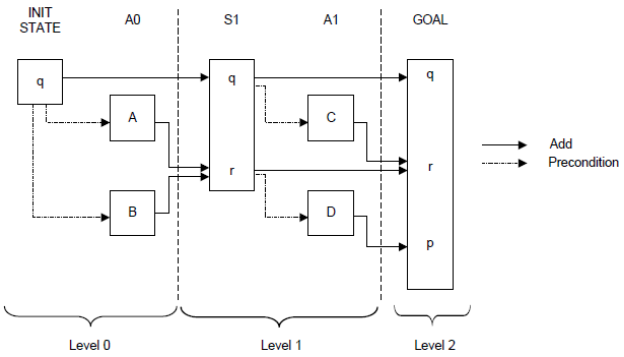


Fig. 2 Relaxed GraphPlan General Structure

The relaxed GraphPlan used as a h_{RG} heuristic consists on building, in any search step, a planning graph from which a solution to the relaxed problem could be extracted. The length of the solution, i.e. the number of actions selected is taken as an estimation of how far the goal is from the current state. HPP like FF, uses this heuristic.

3.2 Extension

The aim of the extension is to provide a tighter communication between the two main components of PIPSS. This means that the planner and the scheduler exchange information in order to guide the search process, thus pruning some parts of the states space. In our first implementation of PIPSS the only information that the planner and the scheduler exchanged was the time and the resources consistency returned by the scheduler. The scheduler could not give any other information back to guide the planner process until the inconsistency was produced. So sometimes we were spending time on searching for a solution that looked in advance we could have known it was inconsistent. Then, the motivation of our work is to estimate the makespan and detect earlier inconsistencies. For this extension we have defined the following terms associated to the relaxed GraphPlan and the search tree.

- The related definitions to the relaxed GraphPlan are:
  - W_{ij} is called the duration of action i at level j. For each level *SumWeight* (SW_j) is defined as:

$$SW_j = \sum_{i=1}^{a_j} W_{ij} \tag{1}$$

Where a_j is the number of actions of level j, and SW_j the sum of all action durations of level j.

– *MacroInitialEstimation (MIE)* is the sum of durations of all the actions of a relaxed GraphPlan (*RG*). It is defined as:

$$MIE(RG) = \sum_{j=0}^{nl} SW_j = \sum_{j=0}^{nl} \sum_{n=1}^{a_j} W_{nj} \tag{2}$$

– *MinimumWeight0 (MinW0)* is the minimum value of the actions durations of the level *j=0*. It is defined as:

$$MinW0(RG) = minW_{i0}, i = 1..a_0 \tag{3}$$

Figure 3 shows an example calculation for the variables W_{ij}, SW_j, *MIE(RG)* and *MinW0(RG)*. Note that every action has an associated duration, taken from the problem, represented by the W_{ij} variable. In level 0 we have $W_{10} = 3$ and $W_{20} = 8$. And for level 2 there are $W_{11} = 7$ and $W_{21} = 5$. In addition, we can see what are the values of the variables SW_j for each level and how it is computed. The values obtained are $SW_0 = 11$ and $SW_1 = 12$. Thus, we can observe that the variable value *MIE(RG)* corresponds to the sum of all the variables SW_j previously calculated. Therefore $MIE(RG) = 23$. Al last, for computing *MinW0(RG)* is observed that there are two actions at level 0, then the *MinW0* value for *RG* will be equal to the $min(W_{10}, W_{11})$ value , i.e. *MinW0(RG)=3*.

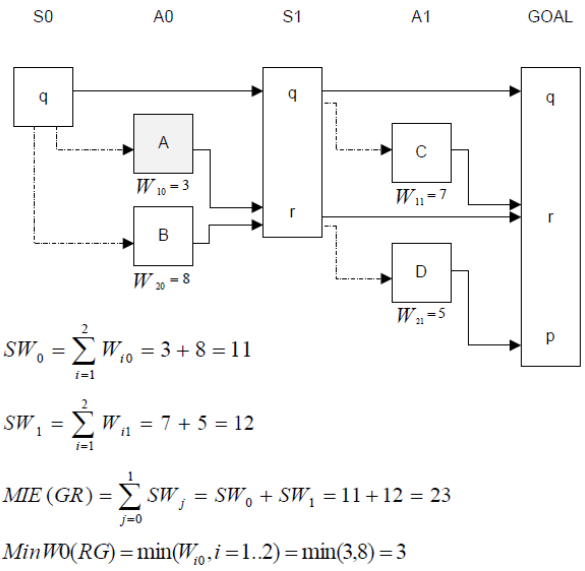


Fig. 3 Example Calculation of Variables *W, SW, MIE* and *MinW0*

- The related definitions to the search tree are:
  - For each tree node V_k (where k is a node identifier) associated to RG_k, where $V_0 = RG_0 = RG$ is the root tree node, the (2) and (3) equalities become true:

$$MIE(V_k) = MIE(RG_k) \tag{4}$$

$$MinW0(V_k) = MinW0(RG_k) \tag{5}$$

  - $MK(V_k)$ is the makespan value returned by the temporal network for the $V_0 - V_{k-1}$ branch. In particular, $MK(V_0) = 0$.

$$MK(V_k) \tag{6}$$

  - *MicroEstimation (mE)* is the sum of the minimum value of the actions durations of the level 0 and the makespan value returned by the scheduler. It is defined as:

$$mE(V_k) = MinW0(V_k) + MK(V_k) \tag{7}$$

    The Figure 4 shows a search tree where the node V_2 is the latest that has been expanded. Suppose that the Figure 3 shows the RG associated with node V_{11}. Then, the value of (5) is 3. Suppose that $MK(V_1) = 6$ then, the value of (7) is 9.

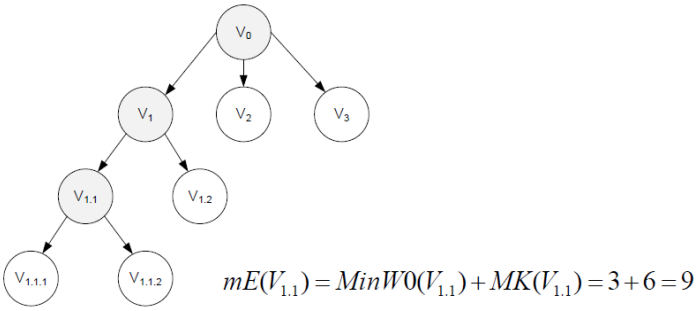


Fig. 4 Example Calculation of Variable *mE*

Next, the concepts previously defined are explained within the system PIPSS*.

Figure 5 shows the algorithm used. Initially we call the function *ExpandNode* that implements a pruning technique to reduce the search space. It takes as a makespan the value of $MIE(V_0) = MIE(RG_0) = MIE(RG)$ if the user does not introduce an initial makespan value. This estimate may not approach the real value because it assumes that the plan is executed sequentially, while in reality it is possible that certain actions can be run in parallel. However, this estimate is used to establish an upper bound. So during the search process those nodes whose value (computed by

the sum of (4) and (6)) exceed $MIE(V_0)$ are not taken into account in the search space. As these branches would guide towards solutions that deviate from the minimum makespan. That is why they are discarded.

Every time a tree node is expanded, the nodes which do not fulfill the mentioned condition are ruled out. Among the nodes that do fulfill it (saved in a list ordered by decreasing value of h_{RG}), the node with the best h_{RG} heuristic is selected (using *getBestNode*).

When this happens, it generates an incomplete partially ordered plan from an incomplete totally ordered plan, which will be sent to the TN in order to find a temporal- and resource- consistent solution. For this partial plan an estimate of the makespan is calculated through the *CalculatedES* function. We do this using (7). The reason to compute in this way the estimation is because the plans sent to the temporal network are partial plans. That is, in every iteration a new operator extracted from level 0 is included in the partial plan. As the aim is to minimize the makespan of the solution, the most promising estimation is equal to the sum of (6) and (5).

If the solution returned by the temporal network is consistent, the function *ExpandNode* is called again with the successor with the best h_{RG} value. Otherwise, the function selects the next successor.

```
Function EstimateMk (cNode,openL,Mk)
-------------------------------------------------------------
cNode the actual state of the problem
mE MicroEstimation value of cNode
bNode the best heuristic succesor of cNode
openL set of nodes that belong to the search space, initially empty
P a plan, initially empty
MK(V_k) makespan value returned by the temporal network
Mk cNode makespan value
TN temporal network
-------------------------------------------------------------
1. openL = ExpandNode(cNode)
2. bNode = getBestNode(openL)
3. P = getPartialPlan(cNode,P)
4. mE = CalculateES(cNode)
5. If IsConsistent(getTNsolution(P),mE)
   Then
      Mk = MK(cNode)
      EstimateMk(bNode,openL,Mk)
   Else
      EstimateMk(cNode,openL,Mk)
6. return P, totalTime
```

Fig. 5 PIPSS* pseudo-algorithm.

4 Experimental Results

The PIPSS extension has been tested against several participants planners of the International Planning Competition of 2008 (IPC-08)[3]. This section is divided into three subsections, each one corresponding to the domains used for experimental purposes.

The three domains are standard PDDL durative domains. The first is *Satellite*, taken from IPC-02[4]. The other domain is called *PipesWorld*, taken from IPC-06[5]. And the last domain called *Openstacks*, is taken from IPC-08. PIPSS* only supports a subset of PDDL2.1 [11], that is why we have chosen these domains.

Six planners have been used for the experimental evaluation:

- PIPSS*: uses the *A vector*, greedy best-first search, ISES and Temporal Search.
- $SGPlan_6$ [7]: partitions the planning problems into subproblems by parallel decomposition, and it uses Metri-FF as the search algorithm. $SGPlan_6$ is the only planner (from IPC-08) that supports PDDL3 [10].
- Metric-FF [14]: is the numeric version of the FF planner. It does a forward search in the state space and uses the relaxed GraphPlan heuristic that ignores the delete lists. It supports PDDL2.1.
- CPT [8]: is an optimal temporal planner based on POCL [8] and constraint programming with makespan optimization via branch and bound. It supports PDDL2.1.
- DAE1 and DAE2 [9]: hybridizes evolutionary algorithms with classical planning methods by dividing the initial problem into a sequence of subproblems, solving each subproblem in turn, and building a global solution from the subproblem solutions with the CPT planner. Each planner uses different strategies for the description of intermediate goals. The first one, DAE1, uses only the predicates that are still present in the goal of the problem, and the second one, DAE2, uses a subset of all predicates.

All these planners (except PIPSS*) took part of IPC-08 temporal satisficing track whose objective is to minimize the total plan duration (makespan).

For each domain we show a table that represents the makespan for each solved problem. The bold values symbolize the best makespan. The total time given to solve each problem was 1800 seconds and all the planners have been launched under the same platform, Windows XP, and those that only run over Linux have been executed under a virtual machine in the same PC. The computer run with an Intel Core 2 Duo processor (2.27Ghz) and 2Gb of RAM memory. Next, an evaluation about the runtime is explained.

[3] http://ipc.informatik.uni-freiburg.de/

[4] http://planning.cis.strath.ac.uk/competition/

[5] http://zeus.ing.unibs.it/ipc-5/

4.1 Satellite Domain

The objective of the Satellite domain is to collect image data from the earth surface through a network of satellites that cover various zones. Every satellite can perform several actions such as turning, switching on and off on-board instrumentation, taking pictures, and selecting among different modes of calibration. The total number of problems is 36 and the complexity gradually increases the number of satellites and instruments, and the number of directions to perform on them. The first problem begins with a single satellite that can take seven different directions, with one instrument and three calibration modes. The last problem has 9 satellites that can take 204 different directions, 22 instruments and 5 modes of calibration.

Table 1 shows the percentage of problems solved by each planner. $SGPlan_6$ solves the highest number of problems followed by PIPSS*. CPT, DAE1 and DAE2 solve very few problems.

Table 1 Problems solved in the Satellite Domain

Name	SGPLAN$_6$	PIPSS*	METRIC-FF	CPT	DAE1	DAE2
Percentage	86%	50%	44%	14%	14%	14%

Table 2 shows the makespan of the solutions for all the planners. PIPSS* gets better makespan values than the other five until problem 22. From here PIPSS* doesn't solve any more problems while $SGPlan_6$ solves until number 30. Metric-FF solves fewer problems than the other 2. It is important to remark that $SGPlan_6$ was the winner of IPC-08 temporal satisfying track. Planners CPT, DAE1 and DAE2 only solve until problem 7. This is why the values are kept constant after this problem.

Table 2 Values Problems solved in the Satellite Domain

SYSTEM	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22
SGPLAN$_6$	221	236	236	422	392	545	344	406	879	912	864	1434	953	717	837	1200	**869**	803	1667	1728	1889	1615
PIPSS*	-	**128**	**128**	**128**	-	**92**	144	**66**	**100**	**137**	**215**	**180**	**110**	**195**	**242**	-	-	**82**	**196**	**170**	**216**	**329**
METRIC-FF	208	225	225	355	251	262	293	224	-	-	281	-	-	463	-	**687**	-	322	467	-	1408	1536
CPT	**132**	152	152	-	**103**	-	**56**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DAE1	**132**	152	152	-	**103**	-	**56**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DAE2	**132**	152	152	-	**103**	-	**56**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Regarding the runtime, DAE1 and DAE2 have the worst performance. PIPSS* is the slowest of the four other planners compared. CPT is the third best followed by SGPlan and Metric-FF.

4.2 PipesWorld Domain

The PipesWorld domain have a network of full fuel tanks (petrol, diesel and derivatives) and fuel empty tanks. They are all interconnected through a complex pipelines network. In turn, the tanks are grouped into areas or sectors. The purpose of this domain is to transport different fuel types of some tanks to others. The complexity of the problem increases with the increasing number of tanks and pipes, besides the different types of fuel. There are 50 problems in the domain. The first begins with 6 available tanks, 3 areas and 2 pipes. Moreover, the last problem has 30 tanks, 5 zones and 5 pipes. Depending on the used pipe the transport time will be higher or lower.

Table 3 shows the planners along with a percentage, indicating how many problems they solve. The best planner in this case is Metric-FF followed by PIPSS*. DAE2 planner is the worst in these terms.

Table 3 Problems solved in the PipesWorld Domain

Name	METRIC-FF	PIPSS*	SGPLAN$_6$	CPT	DAE1	DAE2
Percentage	54%	40%	20%	12%	12%	6%

Table 4 shows the makespan of the solutions for the CPT, DAE1, DAE2, Metric-FF, PIPSS* and SGPlan$_6$ planners. PIPSS* obtains better makespans than the other five until problem 24. From here PIPSS* solves one more problem while Metric-FF solves 9 more. SGPlan$_6$ only solves 10 problems. The rest of planners solve until problem 6.

Table 4 Values Problems solved in the PipesWorld Domain

SYSTEM	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24
SGPLAN$_6$	6	22	16	16	14	14	14	22	40	46	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PIPSS*	**2**	**5**	**5**	**6**	**5**	**6**	**5**	**7**	**5**	**13**	**12**	12	-	-	20	-	**16**	24	-	-	**4**	**10**	**4**	**11**
METRIC-FF	6	22	16	16	14	14	18	24	46	54	19	**11**	-	**19**	**12**	-	-	**18**	**11**	**21**	10	21	-	-
CPT	6	20	12	12	12	12	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DAE1	3	10	6	**6**	6	**6**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DAE2	3	10	56	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Regarding the runtime, the best execution time is achieved by Metric-FF and with almost the same execution time comes SGPlan$_6$. Then CPT and finally PIPSS*. Here also DAE1 and DAE2 have the worst runtime.

4.3 Openstacks Domain

In the Openstacks domain a manufacturer has a number of orders, each for a combination of different products with the restriction of one product at time. There are 30 problems in the domain. The first problem begins with 5 orders, and the last starts with 31 orders.

Table 5 shows the planners used along with the solved problems in %. In this domain Metric-FF, PIPSS* and SGPlan$_6$ solve all the problems. CPT, DAE1 and DAE2, with the same percentage results, are the worst planners.

Table 5 Problems solved in the Openstacks Domain

Name	METRIC-FF	PIPSS*	SGPLAN$_6$	CPT	DAE1	DAE2
Percentage	100%	100%	100%	13%	13%	13%

Table 6 shows the makespan of the solutions. PIPSS* obtains better makespan than the other two. Although, in the first six problems all the others planners performs better than PIPSS* as problems become more complicated PIPSS* improves significantly the results.

The reason PIPSS* offers greater execution times may be because the temporal restriction imposed for the scheduler is very strict, and repeatedly forces the planner to perform backtrack. But in the three domains PIPSS* gets the better makespans in the problem solved.

Table 6 Values Problems solved in the Openstacks Domain

ITEM	SGPLAN$_6$	PIPSS*	METRIC-FF	CPT	DAE1	DAE2
P1	87	230	87	**84**	85	**84**
P2	168	128	157	**114**	127	**114**
P3	170	100	148	**85**	87	**85**
P4	131	133	148	**87**	111	**87**
P5	115	**107**	116	-	-	-
P6	195	**145**	176	-	-	-
P7	168	**108**	112	-	-	-
P8	178	**112**	169	-	-	-
P9	199	**108**	124	-	-	-
P10	214	**110**	214	-	-	-
P11	201	**123**	176	-	-	-
P12	368	**135**	139	-	-	-
P13	318	**146**	223	-	-	-
P14	265	**122**	139	-	-	-
P15	279	**116**	135	-	-	-
P16	288	129	**120**	-	-	-
P17	396	**162**	195	-	-	-
P18	295	**135**	281	-	-	-
P19	305	**159**	195	-	-	-
P20	397	**144**	253	-	-	-
P21	408	**137**	259	-	-	-
P22	432	**146**	197	-	-	-
P23	566	**147**	207	-	-	-
P24	493	**163**	286	-	-	-
P25	441	**205**	211	-	-	-
P26	446	**151**	243	-	-	-
P27	312	**158**	261	-	-	-
P28	507	**183**	216	-	-	-
P29	436	**156**	218	-	-	-
P30	387	**181**	265	-	-	-

5 Conclusions

In this paper we have described PIPSS*, the extension performed on the PIPSS system. PIPSS* has been designed with the purpose of producing a tighter communication between the two systems that is composed of, the planner HPP and the scheduler algorithm ISES.

Results show that PIPSS* performs, in the tested domains, better than the best IPC-08 planner under the temporal satisficing track, SGPlan$_6$. PIPSS* develops a method that estimates, based on temporal heuristic that allows to guide the search process, solutions close to the minimum makespan. This is accomplished by setting certain temporary variables for each tree successor node. Additionally, a makespan restriction is imposed upon the scheduler in order to find a solution that does not exceed the estimated value.

Acknowledgements This work has been founded by the Junta de Comunidades de Castilla-La Mancha project PEII09-0266-6640.

References

1. M. D. R-Moreno and D. Camacho and A. Moreno. HPP: A Heuristic Progressive Planner. The 24th Annual Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG-05). pp: 8-18, London, UK, December, 2005.
2. A. Blum and M. Furst. Fast Planning Through Planning Graph Analysis. Artificial Intelligence, vol. 90, pp: 281-300, 1997.
3. B. Bonet and H. Geffner. Planning as Heuristic Search. Artificial Intelligence, vol. 129, pp: 5-33, 2001.
4. A. Cesta and G. Cortellessa and A. Oddi and N. Policella and A. Susi. A Constraint-Based Architecture for Flexible Support to Activity Scheduling. In Proceedings of the 7th Congress of the Italian Association for Artificial Intelligence on Advances in Artificial Intelligence, pp: 369-381, Bari, Italy, 2001.
5. J. Plaza and M. D. R-Moreno and B. Castano and M. Carbajo and A. Moreno. PIPSS: Parallel Integrated Planning and Scheduling System. The 27th Annual Workshop of the UK Planning and Schedulinmg Special Interest Group (PLANSIG-08). Edinburgh, UK, December, 2008.
6. A. Cesta and A. Oddi and S. F. Smith. An Iterative Sampling Procedure for Resource Constrained Project Scheduling with Time Windows. In Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99). Stockholm, Sweden, 1999.
7. C. W. Hsu and B. W. Wah. The SGPlan Planning System in IPC-6. International Planning Competition 6. Corvallis, OR, USA, 2008.
8. V. Vidal and H. Geffner. Branching and Pruning: An Optimal Temporal POCL Planner based on Constraint Programming. Artificial Intelligence, vol. 170 (3), pp: 298-335, 2006.
9. J. Bibaï and P. Savéant and M. Schoenauer and V. Vidal. DAE: Planning as Artificial Evolution (Deterministic part). International Planning Competition 6. Corvallis, OR, USA, 2008.
10. A. Gerevini and D. Long. Plan Constraints and Preferences in PDDL3. The Language of the Fifth International Planning Competition. Technical Report, Department of Electronics for Automation, University of Brescia, Italy. 2005.
11. M. Fox and D. Long. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. University of Durham, February, Durham, UK, 2002.
12. R. Cervoni and A. Cesta and A. Oddi. Managing Dynamic Temporal Constraint Networks. In Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems (AIPS-94). Chicago, USA, 1994.
13. J. Hoffmann and B. Nebel. The FF Planning System: Fast Plan Generation Through Heuristic Search. Journal of Artificial Intelligence Research, 14, pp: 253-302, 2001.
14. J. Hoffmann. The Metric-FF Planning System: Translating 'Ignoring Delete Lists' to Numeric State Variables. Journal of Artificial Intelligence Research, vol. 20, pp: 291-341, 2003.

Extending SATPLAN to Multiple Agents

Yannis Dimopoulos, Muhammad Adnan Hashmi, Pavlos Moraitis

Abstract Multi-agent planning is a core issue in the multi-agent systems field. In this work we focus on the coordination of multiple agents in a setting where agents are able to achieve individual goals that may be either independent, or necessary for the achievement of a global common goal. The agents are able to generate individual plans in order to achieve their own goals, but, as they share the same environment, they need to find a coordinated course of action that avoids harmful (or negative) interactions, and benefits from positive interactions, whenever this is possible. Moreover, agents are interested in finding plans with optimal length where preference is given to the length of the joint plan. We formalize these problems in a more general way with respect to previous works and present a coordination algorithm which provides the optimal solution in the case of two agents. In this algorithm, agents use μ-SATPLAN as the underlying planner for generating individual and joint consistent plans. This planner is an extension of the well known classical planner SATPLAN, aiming to deal with negative and positive interactions and, therefore, with multi-agent planning problem. Finally we present the experimental results using the multi-agent planning problems from the domains proposed and used in classical planning, which demonstrate the effectiveness of μ-SATPLAN and the coordination algorithm.

1 Introduction

Multi-agent planning is an important issue in the multi-agent systems field. Several works have been proposed in the literature covering different aspects of the problem

Yannis Dimopoulos
University of Cyprus, CY-1678 Nicosia, Cyprus

Muhammad Adnan Hashmi
University Pierre and Marie Curie, 75005 Paris, France

Pavlos Moraitis
University Paris Descartes, 75270 Cedex 06 Paris, France

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_10,

of coordinating the plans of several agents operating in the same environment (see e.g. [1], [2], [3], [4], [5], [6], [7], [8],[9]).

In this paper we study the coordination of multiple agents in a setting where agents are able to achieve individual goals that may be either independent (case of self-interested agents), or necessary for the achievement of a global common goal (case of collaborative agents). In the second case agents have complementary capabilities. It means that they can achieve alone the individual goals which are necessary for the achievement of the global goal which is common to both agents. However none of them has the necessary capabilities to achieve all the goals and therefore the global goal alone.

We assume that in both cases the failure of individual goals (and of the global goal if any) is a worse outcome compared to the achievement of individual goals (and global goal) through suboptimal plans (i.e. plans with greater cost or execution time) and, therefore, suboptimal plans may be considered for adoption by the agents during the planning process. The agents are able to generate and execute their plans independently. However, as they operate in the same environment, conflicts may arise. Therefore, they need to coordinate their course of action in order to avoid *harmful (or negative) interactions* and also to benefit from situations where *positive interactions* might arise. We will call the problem of finding such a pair of plans as *Multi-Agent Coordinated Actions Problem* (MACAP). Moreover, there could be more than one solution to a MACAP and we are interested in finding the optimal one. We will call the problem of finding an optimal solution as *Multi-Agent Optimal Coordinated Actions Problem* (MAOCAP).

Our work extends significantly the work of [10]. More precisely compared to the above work, our work proposes μ-SATPLAN, an extension of famous classical planner SATPLAN to deal with the multi-agent planning problem by handling negative as well as positive interactions. Our work also presents many experimental results using the multi-agent planning problems from the domains proposed and used in classical planning. The experimental results show that our approach is a viable approach to multi-agent planning. It is worth noting that the idea of using satisfiability for solving the closely related problem of plan merging has been considered in [11]. However, in plan merging no new actions can be added in the new plan, i.e. an action cannot belong to the final plan if it does not appear in some of the plans that are merged. Moreover, [11] does not study the parallel encoding of planning into satisfiability that we investigate here.

Another work which extends a classical planner to a multi-agent context is proposed in [12], where they have extended Graphplan [13] for multi-agent planning by proposing Distributed Graphplan. Major difference of their work to ours is that they don't study the positive interactions while coordinating the plans of two agents. Moreover while Distributed Graphplan is only for two agents, μ-SATPLAN finds consistent plans for an arbitrary number of agents. Another major difference is again in the fact that they use the technique of plan merging in which an action can not belong to the joint plan if it is not included in the planning graphs of the individual agents.

The rest of the paper is organized as follows. Section 2 puts some light on the planning as satisfiability framework. Section 3 formally defines the MACAP and MAOCAP. Section 4 presents the solution to the MACAP by presenting μ-SATPLAN, our extension of the SATPLAN for multiple agents. Sections 5 presents the proposed solution for MAOCAP. Section 6 presents the experimental results. Section 7 concludes and discusses future work.

2 Propositional Satisfiability based Planning:SATPLAN

We assume that the agents' planning domain theories are described in the STRIPS language, and denoted by D_α the set of actions that appear in the domain theory of agent α. To generate their plans the agents use μ-SATPLAN, our extension of SATPLAN system [14]. The rationale behind choosing the propositional satisfiability approach to planning is twofold. First, it is one of the most computationally effective approaches to optimal (wrt plan length) STRIPS planning [15, 16]. Second, it can be easily extended to accommodate the needs of our multi-agent planning scenario.

We assume that the reader is familiar with the propositional satisfiability encoding of STRIPS planning. Here we recall very briefly the basics of SATPLAN approach to planning. First, a plan length k is assumed, and the planning problem is translated into a propositional theory (set of clauses). If the resulting satisfiability problem is solvable, a plan of length k is extracted from the model that has been found. Otherwise, the plan length is set to $k+1$ and the procedure iterates.

Among the several ways to transform a planning problem into satisfiability one, we adopt the Graphplan-based parallel encoding [14]. The facts of the (fully specified) initial state and the final state are translated into literals. The propositional theory also contains clauses that constraint actions to imply their preconditions, and fluents to imply the disjunction of all actions that have these fluents in their add-effects. Finally, conflicting actions are declared as mutual exclusive through suitable binary clauses that are added to the theory. For a description of the latest version of SATPLAN, please refer to [16].

In the following we assume that a plan is a set of temporal propositions of the form $A(t)$, where A is an action and t is a time point, meaning that action A executes at time t. If D is a domain theory, I an initial state, P a plan and G a set of goals, the notation $P \models_{D,I} G$ denotes that P is a plan for goal G in the domain D with initial state I, under the standard STRIPS semantics. When there is no possibility for confusion, we simply write $P \models G$.

3 MACAP and MAOCAP

In a multi-agent coordinated actions scenario, a number of agents need to generate individual plans that achieve individual goals which are either independent or necessary for the achievement of a common global goal. We restrict ourselves to the

case of two agents, α and β, and study a scenario that is defined by the following characteristics:

- Each agent is able to achieve his goals by himself. These individual goals may be necessary for the achievement of a global common goal (cooperative setting) that none of the agents can achieve alone. Moreover, agents have different capabilities. In the most extreme case, the effects of the actions of the agents are disjoint.
- Plan length is the criterion for evaluating the quality of both the individual and the joint plans, with preference given to the joint plan length.

The *coordinated actions* problem is defined formally as follows:

Definition 1 (MACAP). *Given two agents α and β with goals G_α and G_β that may be either independent or necessary for a global common goal G_{global} achievement i.e. G_{global}=$G_\alpha \cup G_\beta$, initial states I_α and I_β, and sets of actions D_α and D_β respectively. Find a pair of plans (P_α, P_β) such that*

- $P_\alpha \models_{D_\alpha, I_\alpha} G_\alpha$ *and* $P_\beta \models_{D_\beta, I_\beta} G_\beta$
- P_α *and* P_β *are non-conflicting*

Such pair of plans (P_α, P_β) is called a solution to the MACAP.

We refer to the plans P_α and P_β as *individual* plans, and to the pair (P_α, P_β) as *joint* plan. Moreover, we use the term joint plan to also refer to the plan $P_\alpha \cup P_\beta$. The plan length of a joint plan (P_α, P_β) is defined as $max(l(P_\alpha), l(P_\beta))$.

Definition 2 (MAOCAP). *Given two agents α and β with goals G_α and G_β that may be either independent or necessary for a global common goal G_{global} achievement i.e. G_{global}=$G_\alpha \cup G_\beta$, initial states I_α and I_β, and sets of actions D_α and D_β respectively. Find a pair of plans (P_α, P_β) such that*

- (P_α, P_β) *is a solution to the MACAP for agents α and β.*
- *There is no other solution (P'_α, P'_β) to the problem such that $max(l(P'_\alpha), l(P'_\beta)) < max(l(P_\alpha), l(P_\beta))$.*

In MAOCAP, agents seek to minimize the length of the joint plan, even in the case where this leads to non-optimal individual plans.

4 Solving the MACAP using μ-SATPLAN

To solve the MACAP, we propose a setting in which an agent, say agent α computes his individual plan without taking into account possible conflicts with the plans of other agents. Then this plan is sent to the other agent , say agent β, who computes his plan which does not have any conflict (i.e. *negative interactions*) with the plan of agent α, and which avails the cooperative opportunities (i.e. *positive interactions*)

offered by agent α if such opportunities exist. We will call such plan of agent β as consistent with the plan of agent α.

The *negative interactions* come from two different sources that are discussed below.

1. *Causal link threatening*. This conflict is well known in the context of partial order planning [17]. Let $A_1(t_1)$ and $A_2(t_2)$ be two actions of a plan P such that $t_1 < t_2$ and $A_1(t_1)$ is the latest action of the plan P_1 that adds the precondition p of action $A_2(t_2)$. Then, we say that there is causal link between time points t_1 and t_2 related to p, denoted by the triple (t_1, t_2, p).
   Furthermore, if p is a precondition of an action $A(t)$, p appears in the initial state, and there is no action in plan P that adds p and is executed at some time point $t' < t$, then there is a causal link $(0, t, p)$ in P. Moreover, if $A(t)$ is the last action that adds a goal g, there exists a causal link (t, t_{fin}, g), where t_{fin} is the plan length.
   Finally, if p is a proposition that belongs both to the initial and the final state of planning problem, and there is no action in plan P that adds p, then P contains the causal link $(0, t_{fin}, p)$.
   An action $A(t)$ *threatens the causal link* (t_1, t_2, p) if $t_1 \leq t \leq t_2$ and A deletes p.

2. *Parallel actions interference*. This conflict was introduced in Graphplan [13]. Two actions interfere if they are executed in parallel and one deletes the preconditions or add effects of the other.

The *positive interactions* allow agents to benefit from the effects of actions performed by other agents in order to achieve their own goals. In this situation agent β could use some effects produced by agent α's actions and therefore avoiding to establish facts that have already been established by agent α.

In the following we will present how agents α and β compute their plans using μ-SATPLAN.

4.1 Independent Plan Computation

Agent α needs to compute his plan and also a set of causal links to provide them to other agents so that no agent threatens them. For this purpose, agent α invokes μ-SATPLAN using the call $ComputeNewPlan(T_\alpha, G_\alpha, L, P_\alpha, C_{P_\alpha})$, where T_α includes the agent's domain theory and initial state, G_α is the set of goals of the agent and L is an upper bound on the length of the generated plan (i.e. if $l(P_\alpha)$ is the length of the generated plan, $l(P_\alpha) < L$ holds). This call returns a plan P_α for the achievement of all the goals or returns $fail$ in the argument P_α. This call also returns the set of causal links C_{P_α} of plan P_α. μ-SATPLAN uses the SATPLAN planner intact to compute plan P_α and calculates the set of causal links C_{P_α} by using the Algorithm 1. This algorithm calculates a set C_P consisting of all the causal links of an input plan P.

Algorithm 1 Computing Causal Links

```
C_P ← ∅
for Every level i from goal level going back to level 1 do
  for Every action a at level i do
    for Every precondition p of action a do
      Search in previous layers the latest action, which adds fact p
      if Found an action at level k, which adds fact p then
        Add causal link (k, i, p) to the set C_P
      end if
      if No action found which adds fact p then
        Add causal link (0, i, p) to the set C_P
      end if
    end for
  end for
end for
for Every goal fact g do
  for Every level i starting from goal level going back to level 1 do
    for Every action a at level i do
      Search in previous layers the latest action, which adds fact g
      if Found an action at level k which adds fact g then
        Add causal link (k, goal level, g) to the set C_P
      end if
      if No action found which adds fact g then
        Add causal link (0, goal level, g) to the set C_P
      end if
    end for
  end for
end for
```

4.2 Coordinated Plan Computation

Agent β receives a plan P_α and a set of causal links C_{P_α} from agent α and computes a plan P_β which is consistent with P_α by invoking μ-SATPLAN using the call $ComputeCoordinatedPlan(T_\beta, G_\beta, P_\alpha, C_{P_\alpha}, P_\beta)$, where T_β includes the agent's domain theory and initial state, G_β is the set of goals of the agent. To compute such a plan we have proposed following solutions for handling negative and positive interactions.

4.2.1 Handling Negative Interactions

As we have discussed earlier, negative interactions come either from *causal link threatening* or from *parallel actions interference*. Here we explain how to deal with *causal link threatening*.

While constructing his planning graph, agent β checks before putting any operator O at action level i, if it threatens any of the causal links (t_1, t_2, p) from the set C_{P_α} where $t_1 \leq i \leq t_2$. If so then agent β does not put operator O in the planning graph at level i, even though all of its preconditions are satisfied at this level. Agent β

then expands his planning graph by adding new levels in order to reach all his goals and if the problem of threat persists he avoids to use operator O. In this situation operator O is abandoned.

Example 1 *In order to illustrate, let's consider the well known domain of Blocks World. Let* $ON(X,Y)$ *be a fact meaning that block* X *is on block* Y *and* $MOVE(X,Y,Z,i)$ *be an operator meaning that block* X *moves from* Y *onto* Z *at time* i*. Now let's consider that set* C_{P_α} *has a causal link* $(1,3,ON(a,b))$ *and agent* β *is checking the applicability of the operator* $MOVE(a,b,c,1)$ *in his planning graph. According to our proposal he decides not to put this operator at this level because it threatens a causal link of agent* α*. Moreover he does not put this operator at levels* 2 *or* 3*. Actually* $(1,3,ON(a,b))$ *causal link means that block a should be on block b from time* 1 *to time* 3 *because it is needed by agent* α *at time* 3*. If agent* β *moves block a from b onto c between time points* 1 *and* 3 *then it would spoil the plan of agent* α*.*

We will discuss how to deal with *parallel actions interference* in the following section, where we will also discuss how we deal with *positive interactions*.

4.2.2 Handling Positive Interactions and Parallel Actions Interference

For handling positive interactions, we made the following changes in the classical SATPLAN which are included in μ-SATPLAN.

When agent β starts computing his plan, μ-SATPLAN creates an action for each time step i in the plan of agent α. It means that if there are n time steps in the plan of agent α, it creates n actions $NONAME$ namely $NONAME(0)$, $NONAME(1)$, $NONAME(2)$,...,$NONAME(n)$ such that:

- The add effects of action $NONAME(i)$ are all those facts added by agent α at time i.
- The delete effects of action $NONAME(i)$ are all those facts deleted by agent α at time i.
- The preconditions of action $NONAME(i)$ are all those facts that are preconditions of the actions of agent α at time i.

Thus an action $NONAME(i)$ represents all the actions in the plan of agent α, which are executed in parallel at time i. Agent β while constructing his planning graph, explicitly puts the action $NONAME(i)$ at action level i. So it is obvious that proposition level i has now all the facts added or deleted by agent α in his plan at time i. By doing this, agent β is maintaining the information about the facts added and deleted by agent α at each level.

As it is known the planning graph is then encoded into a CNF format sentence and the solver tries to find a truth value assignment for this sentence. This truth value assignment is actually the solution to the planning problem. Here an important issue arises. The purpose of adding $NONAME$ actions in the planning graph of

agent β is to make sure that agent keeps himself up-to-date with the changes made in the environment by agent α, so we also have to ensure that the solver necessarily choose these *NONAME* actions in the solution it finds, otherwise it would not fulfil the purpose. To ensure this, μ-SATPLAN explicitly adds these *NONAME* actions as unary clauses in the CNF sentence. Thus the solver now tries to find a solution including all the *NONAME* actions. This approach has two advantages. The first one is that we deal with *positive interactions* and the second one is that we simultaneously deal with the *parallel actions interference*. In fact as we have added the actions of agent α's plan in the planning graph of agent β, if one action of agent β at level i is in interference with an action of agent α at this level, then these actions are automatically declared as mutually exclusive by the planning graph mechanism of μ-SATPLAN. As it is known, if a pair of actions is considered mutually exclusive then both actions can not be executed in parallel. So the solver does not select the action of agent β at level i, which interferes with some action of agent α at this level. We illustrate our approach by the following example.

Example 2 *Consider two agents α and β. Agent α has already computed his plan which is $P_\alpha = \{A1(0), A2(0), A3(1)\}$ The positive effects of the actions of P_α are $eff(A1) = a0$, $eff(A2) = a1$, $eff(A3) = a2$. Agent α sends this information to agent β. Agent β creates two NONAME actions because there are two time steps in the plan of agent α. The add effects of $NONAME(i)$ are all the effects added by agent α at time i. So in this case $eff(NONAME(0)) = \{eff(A1) \cup eff(A2)\} = \{a0, a1\}$ and $eff(NONAME(1)) = \{eff(A3)\} = \{a2\}$. The domain theory of agent β contains the actions $D_\beta = \{B1, B2, B3, B4\}$ with the following preconditions and positive effects namely $prec(B1) = \{a6\}$, $eff(B1) = \{a5\}$, $prec(B2) = \{a5\}$, $eff(B2) = \{a0\}$, $prec(B3) = \{a5\}$, $eff(B3) = \{a7\}$, $prec(B4) = \{a0, a7\}$, $eff(B4) = \{a8\}$. The goal of agent β is $G_\beta = \{a2, a4, a7, a8\}$. Thus agent β creates his planning graph and adds all actions $NONAME(i)$ at action level i. Planning graph of agent β is shown in Figure 1 (Gray lines are NOOPS. Boxes are showing the actions. Small letters followed by numbers are propositions. A line from proposition F to an action O shows that F is the precondition of O. A line from an action O to a proposition F shows that F is add effect of O.) In this figure we can see that $a0$ is needed by $B4$ to produce $a8$ and there are two actions which add $a0$ namely $NONAME(0)$ and $B2$. So the solver has to choose between $NONAME(0)$ and $B2$, when this planning graph is converted into a CNF format sentence. However we have to make sure that the solver necessarily choose the former because otherwise it would mean that effect $a0$ which has already been added by agent α, would be added again by agent β. The solver chooses $NONAME(0)$ because we have explicitly added $NONAME(0)$ and $NONAME(1)$ as unary clause in the CNF sentence and now CNF sentence cannot be made true without assigning a true value to the $NONAME(0)$ and $NONAME(1)$. Agent β's plan will be $P_\beta = \{B1(0), B3(1), B4(2)\}$. It is therefore clear from the plan that agent β does not re-establish the fact $a0$ which has already been established by agent α.*

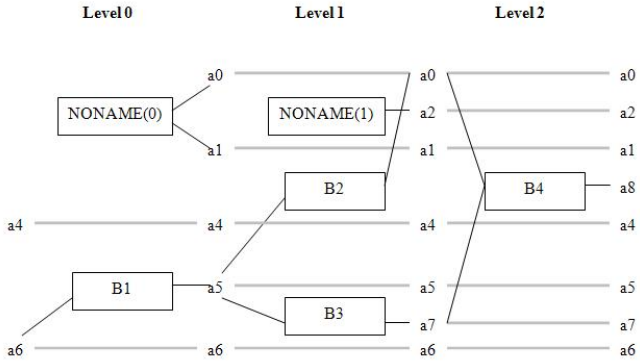


Fig. 1 Illustrating Positive Interactions.

4.3 Coordinated Plan for n^{th} agent

μ-SATPLAN is capable of solving MACAP for n agents where n could be any number greater than one. It means that μ-SATPLAN can be used for finding a plan for the n^{th} agent when $n-1$ agents have already computed their non conflicting plans. Suppose that $n-1$ agents have generated their consistent plans P_1, P_2,..., P_{n-1}. Suppose also that the sets of their causal links are C_1, C_2,..., C_{n-1}. Then a call $ComputeCoordinatedPlan(T_n, G_n, P_1 \cup P_2 \cup .... \cup P_{n-1}, C_1 \cup C_2 \cup .... \cup C_{n-1}, P_n)$ returns a plan P_n for the n^{th} agent which is consistent with the plans of $n-1$ agents.

5 Solving the MAOCAP

To solve the MAOCAP for two agents α and β with goals G_α and G_β, domain theories D_α and D_β and initial states I_α and I_β respectively, we propose a coordination algorithm (Algorithm 2). Each agent uses the μ-SATPLAN for plan generation, and exchanges messages with the other agent.

First agent α computes his plan P_α using μ-SATPLAN by call *ComputeNewPlan* and sends to agent β as a candidate sub-plan of a joint plan. Then agent β computes a plan P_β consistent with P_α using μ-SATPLAN by call *ComputeCoordinatedPlan* and sends the joint plan (P_α, P_β) to agent α. At this point the joint plan (P_α, P_β) becomes the best current joint plan and now it's agent β's turn to compute and propose a candidate sub-plan, which is then processed by agent α. In this way agents take turns to generate and propose candidate sub-plans which are then processed by the other agent to compute a joint plan. Every time, a joint plan found whose length is less than the current best joint plan, becomes the current best joint plan. This way both agents work out to find an optimal joint plan.

The agents exchange messages of the form (P_1, P_2), where P_1 and P_2 are (possibly empty) individual plans. The coordination algorithm (Algorithm 2) refers to agent β and describes how these messages are processed by the agents. The messages can be of three different types, each carrying a different meaning. They are either of the type (P_1, P_2), or $(P_1, \emptyset)$, or $(\emptyset, \emptyset)$, where P_1 and P_2 are non-empty plans. The meaning of each of these messages, and the reaction of the agents to these messages, are described in the following.

Before moving to the main body of the algorithm, the agents go through a phase in which the algorithm's variables and data structures are initialized. Moreover, agent α sends a message of the form $(P, \emptyset)$, where P is the (optimal) plan generated by the call $ComputeNewPlan(T, G, \infty, P, C)$, where T and G are the agent's domain theory and goals respectively.

Each incoming message is processed by the coordination algorithm in a way that depends on its type. A message of the form $(P, \emptyset)$, means that the other agent proposes P as a candidate sub-plan of a joint plan. The set of causal links C is also sent along with the plan. But, to simplify the presentation, we are not showing it explicitly here. The receiving agent checks, by invoking μ-SATPLAN as explained earlier, if he can generate a plan P' that achieves his own goals and is consistent with P. An additional requirement is that the length of the joint plan $P \cup P'$, defined as $max(l(P), l(P'))$, is shorter than the best joint plan. If this is the case, the agent sends the message (P, P') to the other agent, meaning that P can be a part of an improved joint plan (P, P'). If the agent that receives the message $(P, \emptyset)$ fails to find a plan as specified above, he sends the message $(P, fail)$, indicating that P cannot be part of a better joint plan. Then, the agent attempts to generate a new sub-plan with length shorter than l_{best}. If such a sub-plan exists, he sends it to the other agent. Otherwise, he sends the message $(\emptyset, \emptyset)$, indicating that there are no shorter individual plans.

A message of the form (P_1, P_2), with $P_1 \neq \emptyset$ and $P_2 \neq \emptyset$, is a reply to an earlier message, where he proposed the plan P_1 to the other agent. Upon processing such a message, if $P_2 \neq fail$ and the proposal (i.e. plan P_2) leads to an improved joint plan (P_1, P_2), the variables P_{best} and l_{best} are updated accordingly but if $P_2 \neq fail$ then agent deletes P_1 from memory because this plan does not lead to an improved joint plan so it does not need to be stored any more. It ensures that at any point there are at most only two plans stored, one which is under consideration and one which is part of the current optimal. Then if possible, a new candidate sub-plan is generated and sent to the other agent. Otherwise a message $(\emptyset, \emptyset)$ is sent.

Upon receiving a message $(\emptyset, \emptyset)$, an agent sets his *expect* variable to false, meaning he does not expect any further candidate sub-plans from other agent. If *continue* variable is true, it generates another plan and sends to other agent, otherwise comes out of the algorithm.

The algorithm terminates when the condition $(not\ continue) \wedge (not\ expect)$ becomes true. In such a case, the agent has received replies to all the sub-plans that he has proposed, he has no other plan to propose, and he does not expect any further proposals from the other agent.

Algorithm 2 Coordination Algorithm

while *true* **do**
get_incom_message(P_α, P)
if $P_\alpha \neq \emptyset$ and $P = \emptyset$ **then**
  $ComputeCoordinatedPlan(T_\beta, G_\beta, P_\alpha, C_{P_\alpha}, P_\beta)$
  if $P_\beta \neq fail$ and $max(l(P_\alpha), l(P_\beta)) < l_{best}$ **then**
    $l_{best} := max(l(P_\alpha), l(P_\beta))$, $P_{best} := (P_\alpha, P_\beta)$
    send message (P_α, P_β)

  else
    send message $(P_\alpha, fail)$
  if (*continue*) **then**
    Call Sub-Procedure *New_Proposal*
  if (*not continue*) and (*not expect*) **then**
    exit(P_{best})
else
if $P_\alpha \neq \emptyset$ and $P \neq \emptyset$ **then**
  if $P \neq fail$ **then**
    $l_{best} := max(l(P_\alpha), l(P))$, $P_{best} := (P_\alpha, P)$
  else
    Delete P from memory
  if *continue* and (*not expect*) **then**
    Call Sub-Procedure *New_Proposal*
  if (*not continue*) and (*not expect*) **then**
    exit(P_{best})
else (ie. $P_\alpha = \emptyset \wedge P = \emptyset$)
  if (*not continue*) **then**
    exit(P_{best})
  if (*continue*) **then**
    expect= false
    Call Sub-Procedure *New_Proposal*
    exit(P_{best})

Sub-Procedure 1 *New_Proposal*

  $ComputeNewPlan(T_\beta, G_\beta, l_{best}, P_\beta, C_{P_\beta})$
  if $P_\beta \neq fail$ **then**
    send message $(P_\beta, \emptyset)$
  else
    continue= false
    send message $(\emptyset, \emptyset)$

6 Experimental Results

In this section we present some preliminary experimental results. An important point to note is that, although the coordination algorithm presented in section 5, is sound and generic to find the global optimal solution to the problem of MACAP, in some cases it could become inefficient. So in our current implementation, in order to re-

No	Problem	Number of Goals A/B	Time for First Plan	Length Pairs of First Plan	Best Length Pairs	Joint Plans Computed	Time for Best Plan
1	AIPSLog11	5/4	3	(9,14)	(13,10)	7	16
2	AIPSLog15	4/7	4	(10,17)	(15,10)	10	28
3	AIPSLog18	4/4	5	(10,14)	(11,11)	3	12
4	AIPSLog20	5/5	7	(14,18)	(14,18)	8	39
5	AIPSLog24	5/4	3	(12,14)	(13,12)	5	14
6	AIPSLog28	5/5	7	(10,16)	(15,13)	7	36
7	AIPSLog30	4/5	4	(12,16)	(12,16)	10	28
8	Storage10	2/2	12	No Plan	(11,18)	16	64
9	Storage12	2/2	32	No Plan	(9,9)	10	216
10	Storage16	3/3	129	No Plan	(13,8)	16	942
11	TPP11	3/3	4	(13,13)	(13,13)	3	9
12	TPP13	4/4	7	(9,14)	(10,11)	5	22
13	TPP14	4/5	8	(7,13)	(9,10)	4	34
14	TPP15	5/5	12	(9,15)	(11,11)	6	50
15	TPP17	6/6	29	(11,11)	(11,11)	3	65
16	TPP19	7/7	47	(10,12)	(11,10)	2	90
17	TPP20	9/6	58	(12,11)	(12,11)	4	117

Table 1 μ-SATPLAN performance on multi-agent problems.

duce the search space, and at the same time to improve the quality of the global solutions found, we do not try to generate every possible proposal plan of an agent. Instead, in our system, the first plan proposed by an agent to another agent is the local optimal generated by SATPLAN and then every subsequent plan has the length which is one more than the length of the previous proposal plan. It means that if the first proposal plan of an agent has length 6 then the next proposal plan of agent will be of length 7. In this way agent generates proposal plans until the length of its proposal plans reaches the current *lbest*. At that moment, it sets the value of *continue* variable to false. So our current implementation of the coordination algorithm does not guarantee the global optimal solution, but comes up with a good solution, which in many cases is much better than the first solution found.

We run μ-SATPLAN and the coordination algorithm on *multi-agent versions* of the well-known planning domain Logistics as well as the Storage and TPP domains from the 5th International Planning Competition [15]. The TPP (Traveling Purchaser Problem) is a generalization of the Traveling Salesman Problem. A purchaser can buy products from different markets that provide the products in different amounts and prices. Storage domain is about moving a certain number of crates from some containers to some depots by hoists. Inside a depot, each hoist can move according to a specified spatial map connecting different areas of the depot. For more information about the domains please refer to [15]. To obtain the multi-agent version of a problem, we split the goals of the original problem into different sets and assign each goal set to a different agent. All our experiments refer to the case of two agents. We assume that each agent can execute all actions in the domain, and therefore can achieve its goals without assistance from other agents.

Time	Plan of Agent α	Plan of Agent β
0	(DRIVE T1 D1 M1)	[ ]
1	(BUY T1 P1 M1)	(BUY T1 P2 M1)
2	(LOAD P1 T1 M1)	(LOAD P2 T1 M1)
3	(DRIVE T1 M1 D1)	[ ]
4	(UNLOAD P1 T1 D1)	(UNLOAD P2 T1 D1)

Table 2 A plans pair generated by μ-SATPLAN

The results are shown in Table 1. The underlying SATPLAN system used is SATPLAN 2006. All experiments were run on a machine with a 2.80 GHz CPU and 4096 MBs of memory. A time limit of 3600 seconds was used for μ-SATPLAN. Column *Number of Goals A/B* contains pairs of the form a/b where a (b) is the number of goals assigned to agent A (B). The columns *Time for First Plan* and *Length Pairs of First Plan* provide information about the run time (in seconds) and length of the first joint plan found by μ-SATPLAN. More specifically, an entry (a,b) means that in the first joint plan that is found, agent A (B) finds a plan of length a (b). Similar information is provided in columns *Best Length Pairs* and *Time for Best Plan*, but for the best plan found by the system. Finally, the entries under *Joint Plans Computed* are the total number of joint plans computed by the coordination algorithm, before its termination. An entry *No Plan* in the column *Length Pairs of First Plan* means that there was no consistent plan of agent B for the first plan proposed by agent A.

To understand the efficacy of positive interactions we consider a pair of plans generated by μ-SATPLAN (Table 2). The problem under consideration is from TPP domain. There is one market M1 and one depot D1. Moreover there are two trucks T1 and T2 in the world. M1 is selling products P1 and P2. The goal of agent α is to buy and store P1 in D1 and the goal of agent β is to buy and store P2 in D1. We can see from the plan generated by agent β that he does not utilize truck T2 instead he avails the cooperative oppurtunities offered by agent α by using the same truck T1. At time 0, agent β sits idle just waiting for agent α to move T1 to M1. At times 1 and 2, agent β also buys and puts his product P2 in T1 along with agent α. Then again at time 3 agent β sits idle waiting for agent α to drive T1 from M1 to D1. At time 4, both agents unload and store their products in D1.

The preliminary experimental results of Table 1 show that μ-SATPLAN represents a viable approach to the problem of multi-agent planning.

7 Conclusion and future work

In this paper we formailzed the Multi-Agent Coordinated Actions Problem (MACAP) and Multi-Agent Optimal Coordinated Actions Problem (MAOCAP). We presented μ-SATPLAN, a multi-agent version of SATPLAN the most powerful planner in classical planning, which is used by the agents to solve the MACAP

and MAOCAP. We presented all the details of how μ-SATPLAN deals with negative as well as positive interactions to find consistent plans for multiple agents working in the same environment. Moreover, in this paper we presented, for the first time in multi-agent planning domain, several experimental results that show the added value of adapting SATPLAN for multi-agent planning. We believe that presenting these results is an important issue because it will give the opportunity to other researchers working in multi-agent planning to use these domains and to compare the performance of their planners with μ-SATPLAN.

As in our current implementation, we can not guarantee the global optimal solution, so currently we are investigating the use of heuristics to guide the search of the coordination algorithm, in order to ensure the optimality of the global joint plan and at the same time not to generate a lot of sub-optimal plans before coming up with the optimal one.

References

1. Boutilier, G., Brafman, R.: Partial-order planning with concurrent interacting actions. Journal of Artificial Intelligence Research **14** (2001)
2. Katz, M.J., Rosenschein, J.S.: The generation and execution of plans for multiple agents. Computers and Artificial Intelligence **12**(1) (1993) 5–35
3. Ephrati, E., Rosenschein, J.: Divide and conquer in multi-agent planning. In: AAAI94. (1994)
4. Tsamardinos, I., Pollack, M., Horty, J.: Merging plans with quantitative temporal constraints, temporally extented actions and conditional branches. In: AIPS00. (2000)
5. Tonino, H., Bos, A., Weerdt, M.D., Witteveen, C.: Plan coordination by revision in collective agent-based systems. Artificial Intelligence **142,2** (2002) 121–145
6. Cox, J., Durfee, E.: Discovering and exploiting synergy between hierarchical planning agents. In: AAMAS03. (2003)
7. Cox, J., Durfee, E.: An efficient algorithm for multiagent plan coordination. In: AAMAS05. (2005)
8. Steenhuisen, J., Witteveen, C., Mors, A., Valk, J.: Framework and complexity results for coordinating non-cooperative planning agents. Lecture Notes in Computer Science **4196** (2006) 98
9. Ottens, B., Faltings, B.: Coordinating agent plans through distributed constraint optimization. In: 18th International Conference On Automated Planning And Scheduling, ICAPS. (2008)
10. Dimopoulos, Y., Moraitis, P.: Multi-agent coordination and cooperation through classical planning. In: Proceedings of the IEEE/WIC/ACM Intern. Conf. on Intelligent Agent Technology (IAT). (2006) 398–402
11. A.Mali: Plan merging and plan reuse as satisfiability. In: 5th European Conference on Planning, ECP99. (1999)
12. Iwen, M., Mali, A.: Distributed graphplan. In: Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI2002). (2002) 138–145
13. Blum, A., Furst, M.: Fast planning through planning graph analysis. Artificial Intelligence Journal **90**(1-2) (1997)
14. Kautz, H., McAllester, D., Selman, B.: Encoding plans in propositional logic. In: Principles of Knowledge Representation and Reasoning, KR. (1996)
15. Gerevini, A.: 5th international planning competition: Results of the deterministic truck. Available from http://zeus.ing.unibs.it/ipc-5/
16. Kautz, H., Selman, B., Hoffmann, J.: SATPLAN: Planning as satisfiability. In: Booklet of the 2006 International Planning Competition. (2006) Available from http://zeus.ing.unibs.it/ipc-5/publication.html.
17. Weld, D.: An introduction to least commitment planning. AI Magazine **15**(4) (1994)

MACHINE LEARNING

A New Approach for Partitional Clustering Using Entropy Notation and Hopfield Network

Vahid Abrishami[1], **Maryam Sabzevari**[2] **and Mahdi Yaghobi**[3]

Abstract This paper proposes a new clustering algorithm which employs an improved stochastic competitive *Hopfield* network in order to organize data patterns into natural groups, or clusters, in an unsupervised manner. This *Hopfield* network uses an entropy based energy function to overcome the problem of insufficient understanding of the data and to obtain the optimal parameters for clustering. Additionally, a chaotic variable is introduced in order to escape from the local minima and gain a better clustering. By minimizing the entropy of each cluster using *Hopfield* network, we achieve a superior accuracy to that of the best existing algorithms such as optimal competitive *Hopfield* model, stochastic optimal competitive *Hopfield* network, *k*-means and genetic algorithm. The experimental results demonstrate the scalability and robustness of our algorithm over large datasets.

1 Introduction

Many problems in data analysis require partitioning of data items into a set of clusters in a way that items within a cluster are more similar to each other than they are to items in the other clusters. Partitioning data into a set of clusters plays an important role in a wide variety of applications such as data mining, image segmentation, signal compression and machine learning.

By focusing on some of the discriminating criteria, the clustering algorithms fall into two major categories, the hierarchical and the partitional algorithms. Partitional algorithms are less expensive in both time and space complexity, and

1 Young Researchers Club (YRC), Islamic Azad University, Mashhad Branch
vahidabrishami2005@gmail.com

2 Islamic Azad University, Mashhad Branch
sabzevary.maryam@gmail.com

3 Islamic Azad University, Mashhad Branch
yaghobi@mshdiau.ac.ir

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_11,

therefore, are more popular than hierarchical techniques in pattern recognition. We concentrate on partitional clustering in this paper.

The partitional clustering aims to directly obtain a single partition of the collection of items into clusters base on a criterion function. For instance, the well known *k*-means algorithm [1] is one of these methods which considers the squared error criterion and therefore works well just for hyper spherical data. Bezdek et al. [2] introduce a Fuzzy version of *k*-means algorithm which can model situations where clusters actually overlap by avoiding local minima of the cost function. Sarafis et al. [3] applied GA in the context of *k*-means objective function. A set of *k*-means are considered as a population. This population is improved through the genetic operations. In this method clusters can have different size and elongation but the shapes are limited. A novel method which uses the notation of entropy to group data items is proposed by Barbara et al. [4]. First they heuristically find a set of initial clusters and then greedily add points to the clusters according to a criterion that minimizes the whole entropy of the system. Recently Galan-Marın et al. [5], proposed an optimal competitive Hopfield model (OCHOM) that always guarantees and maximizes the decrement of any Lyapunov energy function. The OCHOM's solution is always valid and it greatly reduces the search space. However, there is no mechanism for escaping from local minima. Wang and Zhou [6] propose a stochastic optimal competitive Hopfield model (SOCHOM). They introduce a hill-climbing dynamics which helps the network to escape from local minima. Nevertheless, they ignore the nature of the data and rely on distance metric. The random noise that is used to escape from local minima is not also so effective.

In this paper we explore an alternative technique based on a novel integration of both entropy and competitive Hopfield network in order to partition data into clusters. The proposed algorithm employs an alternative version of SOCHOM [6] energy function which considers the minimization of entropy in each cluster instead of minimization of distance from the center of a cluster. By using entropy based energy function, some of the problems that are mentioned for SOCHOM [6] algorithm can be solved. Additionally, we have introduced a chaotic variable which is more effective for solving quadratic minimization than the hill-climbing method. Simulation results through several benchmark datasets [10] demonstrate the performance of our algorithm.

2 SOCHOM Method

In order to fix the context and to clarify prolific terminology, consider dataset X consisting of n objects or patterns in a D-dimensional metric space. The ideal goal of clustering is to assign objects to a finite system of K subsets. If the clusters are

represented by $G = \{g_1, g_2, ..., g_k\}$, and $n_k = |g_k|$ is the cardinal of the cluster g_k, then

$$g_k \neq \varnothing \quad For \quad k = 1, 2, 3 \ldots K$$
$$g_i \cap g_j = \varnothing \quad For \quad i \neq j \quad i, j = 1, 2, 3 \ldots K$$
$$\sum_{k=1}^{K} n_k = n.$$

There may be other feasible solutions but the optimal solution G* should be found.

Recently, Galán-Marín et al. [5] proposed a discrete Hopfield model, named OCHOM, which consists of n disjoint groups of *m* neurons. The input, output and bias of the *i*th neuron in the *x*th group is defined as $v_{xi}(t) \in \{0,1\}$, u_{xi} (t), and bias Θ_{xi}. $\omega_{xi,yj}$ demonstrates the interconnection strength between neurons x_i and y_j. The Hopfield energy function that is employed by OCHOM [5] is given by (1):

$$E(t) = -\frac{1}{2}\sum_{x=1}^{n}\sum_{i=1}^{m}\sum_{y=1}^{n}\sum_{j=1}^{m}\omega_{xi,yj}v_{xi}v_{yj} + \sum_{x=1}^{n}\sum_{i=1}^{m}\theta_{xi}v_{xi} \tag{1}$$

and the input of the neurons is defined as below (2):

$$u_{xi}(t) = -\frac{\partial E}{\partial v_{xi}} = \sum_{y=1}^{n}\sum_{j=1}^{m}\omega_{xi,yj}v_{yj}(t) - \theta_{xi}. \tag{2}$$

The energy difference is calculated by (3):

$$\Delta E(t)_x = E(t+1) - E(t) = u_{xo}(t) - (u_{xc}(t) - K_{xo,xc}) \tag{3}$$

where *xo* is the fired neuron for group *x* with the output 1 at time t and x_c is the candidate neuron for group *x* which will be fired at time t + 1 and $K_{xo,xc}$ = -1/2 ($\omega_{xo,xo} + \omega_{xc,xc} - 2\ \omega_{xo,xc}$). The candidate neuron x_c in each group is defined as the neuron with maximum value of u_{xi} (t) – $K_{xo,xi}$. So the input/output function of the *i*th neuron in the *x*th group is given by following (4):

$$v_{xi}(t+1) = 1 \quad If \quad u_{xi}(t) - K_{xo,xi} = \max_{j=1\ldots m}\{u_{xj}(t) - K_{xo,xj}\}; 0 \ \ Otherwise. \tag{4}$$

The structure of OCHOM is displayed in Fig. 1. As it can be seen in Fig. 1, there are *n* (number of data items) groups of *m* (number of clusters) neurons. The input of each neuron is updated by (2) and its related output is calculated using (4).

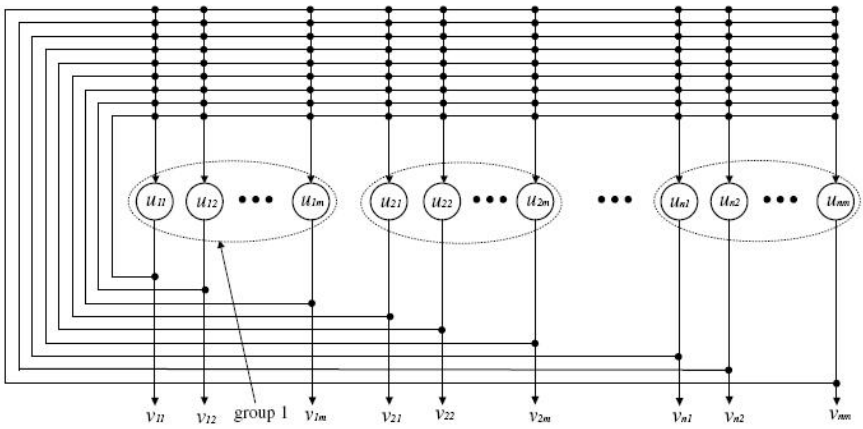


Figure 1 Structure of OCHOM

In (4) there is no mechanism in order to escape from the local minima. Recently, Wang and Zhou [6] propose a stochastic optimal competitive Hopfield network, named SOCHOM, which employs a hill-climbing dynamics to escape from local minima. In their proposed algorithm [6], the input/output function of the *i*th neuron in the *x*th group is modified as (5):

$$u'_{xi}(t) = \alpha(s)(u_{xi}(t) - K_{xo,xi}) \ \ and \ \ v_{xi}(t+1) = 1$$
$$If \ \ u'_{xi}(t) = \max_{j=1\ldots m}\{u'_{xj}(t)\};0 \ \ Otherwise \tag{5}$$

where *s* is the updating step number and u'_{xi} (t) is the transient variable. $\alpha(s)$ is given by (6):

$$\alpha(s) = random(h(s),1) \tag{6}$$

in which $h(s) = 1 - 2e^{-s/\lambda}$.

By considering each data item or object in a dataset *X* as a vector of dimension *D*, the energy function based on SOCHOM [6] is calculated by (7):

$$E = \sum_{k=1}^{K} \sum_{i=1}^{n} v_{xi} \sum_{d=1}^{D} (x_{id} - c_{kd})^2 \tag{7}$$

where x_{id} (i = 1, 2,. . . ,n, d = 1, 2,. . . ,D) is the dth feature of object x_i and c_{kd} is the dth dimension of the kth cluster's center which can be obtained by (8):

$$c_{kd} = \frac{\sum_{i=1}^{n} v_{ik} x_{id}}{\sum_{i=1}^{n} v_{ik}}. \tag{8}$$

According to (7), they obtain the following input updating rule (9) for the clustering problem [6]:

$$u_{xi}(t) = -2\alpha(s). \sum_{d=1}^{D} (x_{id} - c_{kd})^2. \tag{9}$$

The input/output function of the kth neuron in the ith group for the clustering problem is given by (10):

$$v_{ik}(t+1) = 1 \; \textit{If} \; u_{ik}(t) = \max_{l=1\ldots K} \{u_{il}(t)\}; 0 \quad \textit{Otherwise.} \tag{10}$$

In update step of cluster center, if an object i moves from cluster p to q, only the center of these two clusters should recalculated as (11):

$$C_p = \frac{n_p c_p - x_i}{n_p - 1} \qquad C_q = \frac{n_q c_q - x_i}{n_q + 1}. \tag{11}$$

Neural mapping of the partitional clustering problem to the SOCHOM is described by Fig. 2. In Fig. 2 each row demonstrates a group of K (number of clusters) neurons. The black squares in each row means that object i belongs to cluster k or simply v_{ik} = 1.

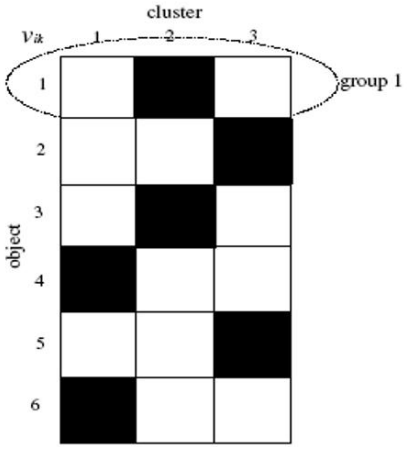


Figure 2 Neural mapping of cluster problem to SOCHOM

3 Algorithm Descriptions

We argue that the SOCHOM algorithm [6] just considers the distance of each data item from the center of each cluster in its energy function and tries to minimize this distance in each cluster. Relying on the usage of a distance metric and ignoring the nature of the data in a cluster can raise some problems. For instance, there is no solution for a data item with the same distance from two different clusters. In this section a novel method is represented which uses the notation of entropy in order to partition data into clusters.

Entropy is the amount of information which is obtained by observing one output of the source. When the entropy increases, the uncertainty is being increased too, therefore more information is related to that source. Assume X is a random variable and $S(X)$ is the set of values that X can take. Thus, we can define the entropy $E(X)$ as shown below (12):

$$E(X) = -\sum_{x \in S(X)} p(x) \log(p(x)) \tag{12}$$

$p(x)$ is the probability function of X. The entropy of a multivariate vector $\hat{x} = \{X_1, X_2, ..., X_n\}$ is defined as (13):

$$E(\hat{x}) = -\sum_{x_1 \in S(X_1)} \cdots \sum_{x_n \in S(X_n)} p(x_1, ..., x_n) \log p(x_1, ..., x_n). \tag{13}$$

Entropy is sometimes referred as a measure of the amount of disorder in a system. According to this, in order to partition n data items into K clusters, our proposed algorithm considers n groups of K neurons and employs an entropy based energy function (14):

$$E = -\sum_{k=1}^{K}\sum_{i=1}^{n} v_{ik} * ... \sum_{x_1 \in S(X_1)} ... \sum_{x_n \in S(X_D)} p(x_1,...,x_D)\log p(x_1,...,x_D). \tag{14}$$

The output $v_{ik} = 1$ means that the ith object belongs to the kth cluster. The algorithm assigns each data item to each cluster, and puts a coefficient of calculated entropy as the input of each neuron in a group. Therefore, the following input updating rule (15) for the clustering problem can be obtained:

$$u_{ih} = 2(ChaoticVariable) * ... \sum_{x_1 \in S(X_1)} ... \sum_{x_n \in S(X_n)} p(x_1,...,x_n)\log p(x_1,...,x_n) \tag{15}$$

Only one neuron within each group with minimum entropy is chosen as a candidate and is fired.

The SOCHOM algorithm [6] employs a random variable in (9) in order to escape from local minima. However, it has been confirmed that chaotic noise is more effective for solving quadratic assignment problem and gains a better performance to escape out local minima than random noise [7]. Therefore, we employ the chaotic variable (16), and use the logistic map [7] in order to produce it:

$$x_{k+1} = \mu x_k (1 - x_k) \qquad x_k \in (0,1),\ k = 0,1,2,...,n,\ \mu = 4 \tag{16}$$

x_k is the input variable for the logistic map and μ is a constant. The values of the chaotic variable should be in range (-1, 1). So, the following optimization is done (17):

$$x_k^* = a_i + (b_i - a_i)x_k. \tag{17}$$

In order to eliminate the chaotic variable after some iteration, the variable is multiplied by a ∂ which is obtained by (18):

$$\partial = 1 - \left|\frac{n-1}{n}\right|^k \tag{18}$$

where n is the number of iterations and k is an integer. So the equation (15) is changed as below (19):

$$u_{ik} = 2(x_k^* \partial) \sum_{x_1 \in S(X_1)} \dots \sum_{x_n \in S(X_n)} p(x_1,\dots,x_n) \log p(x_1,\dots,x_n) \tag{19}$$

The pseudo code of our proposed algorithm can be seen below:

/ Start of Initial Step*
For i = 1 to *n* do
  For k = 1 to *K* do
    assign the initial values of u_{ik} (-1,1) set the
    output of one neuron in each group to be 1 and
    the other neurons in the group to be 0 randomly;
  End For
End For
/ End of Initial Step*

/ Calculating the Center of Each Cluster*
For *k* = 1 to *K* do
  For *d* = 1 to *D* do

$$c_{kd} = \frac{\sum_{i=1}^{n} v_{ik} x_{id}}{\sum_{i=1}^{n} v_{ik}}$$

  End For
End For
/ End of Calculating the Center of Each Cluster*

/ Start of the Main Loop*
do
  For *i* = 1 to *n* do

    / Updating the Input of Each Neuron*
    For *k* = 1 to *K* do

$$x_t = 4x_{t-1}(1 - x_{t-1})$$

$$\partial = 1 - \left| \frac{n-1}{n} \right|^k$$

$$u_{ik} = 2(x_k^* \partial) * \dots \sum_{x_1 \in S(X_1)} \dots \sum_{x_n \in S(X_n)} p(x_1,\dots,x_n) \log p(x_1,\dots,x_n)$$

End For
/ End of Updating the Input of Each Neuron*

*/*Obtaining the Output of Each Neuron*
For k = 1 to K do
$v_{ik}(t+1) = 1$ *If* $u_{ik}(t) = \max_{l=1\ldots K}\{u_{il}(t)\}; 0$ *Otherwise*
End For
*/*End of Obtaining the Output of Each Neuron*

Update cluster center
End For
t = t + 1;
until the state of the network reaches an equilibrium state

4 Experimental results

In order to compare the performance of our proposed algorithm with the SOCHOM algorithm, we consider the same PC (Pentium4, 2.8 GHz) and also the same benchmark data sets [10] (Fisher's iris, Wisconsin breast cancer, Ripley's glass, Vowel) as SOCHOM [6]. SE[4] and VRC[5] [8][9] are considered as two criteria for comparing two algorithms. SE is defined as the sum of squared Euclidian distances between each object in a cluster and its cluster center.VRC is sum of squared distances between the cluster centers and the mean vector of all objects. The best, average and worse results over 10 runs which are provided by our algorithm and also SOCHOM algorithm [6] are presented in Table 1.

Table 1. The best, average and worse results

Data Set	Criterion	SOCHOM			Proposed Alg.		
		Worse	Average	Best	Worse	Average	Best
Iris	SE	7885.14	7885.14	7885.14	7885.14	7885.14	7885.14
	VRC	561.63	561.63	561.63	561.63	561.63	561.63
Cancer	SE	19,323	19,323	19,323	19,323	19,323	19,323
	VRC	1026.26	1026.26	1026.26	1026.26	1026.26	1026.26
Glass	SE	338.74	337.17	336.06	337.05	336.62	335.23
	VRC	123.29	124.01	124.62	124.35	124.69	125.19
Vowel	SE	30,690,658	30,686,680	30,686,238	30,687,253	30,686,143	30,685,825
	VRC	1465.64	1465.85	1465.88	1465.71	1465.87	1465.92

[4] Squared error (SE) criterion: sum of squared Euclidean distances between each object in g_k and its cluster center c_k.

[5] Variance ratio criterion (Paterlini and Krink [8], Jarboui et al. [9])

As it can be seen in Table 1, the results of our method for not challenging data sets as Iris and cancer are same as the SOCHOM algorithm [6]. However, for data sets which are strongly overlapped and are of a larger size, our algorithm demonstrates a better performance than SOCHOM. It is because the proposed algorithm considers the nature of data and not just relies on measurement parameters. From all simulation results, we can conclude that our algorithm can find results comparable to or better than the SOCHOM approach [6]. Since, the SOCHOM method has been compared with other methods like optimal competitive Hopfield model, *k*-means and genetic algorithm in [6], and has shown a better performance than these methods, we can conclude that the proposed method shows a superior accuracy to the mentioned methods.

5 Conclusions

In this paper we present a stochastic optimal competitive Hopfield network which employs an entropy based energy function in order to minimize the entropy of each cluster. The proposed algorithm also uses a chaotic variable to escape out local minima and gains a better solution. A comparison between our proposed algorithm and the SOCHOM algorithm [6] confirms the better performance of our algorithm over larger and more overlapping data sets.

References

1. McQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, pp. 281--297 (1967).
2. Bezdek, J. C., Ehrlich, R., & Full, W.: Fcm: The fuzzy c-means clustering algorithm. J. Computers & Geosciences, 10 (2-3), pp. 191--203 (1984).
3. Sarafis, I., Zalzala, A. M., Trinder, P. W.: A genetic rule-based data clustering toolkit. In: Proceedings of the Evolutionary Computation (CEC '02), pp. 1238-1243. IEEE Computer Society, Washington, DC (2002).
4. Barbara, D., Couto, J., Li, Y.: COOLCAT An entropy based algorithm for categorical clustering. In: Proceedings of the eleventh international conference on Information and knowledge management. ACM Press (2002).
5. Galán-Marín, G., Mérida-Casermeiro, E., and Muñoz-Pérez.: Modeling competitive Hopfield networks for the maximum clique problem. J. Comput. Oper. Res. 30, pp. 603--624 (2003).
6. Wang, J. and Zhou, Y.: Stochastic optimal competitive Hopfield network for partitional clustering. J. Expert Syst. Appl. 36, pp. 2072--2080 (2009).
7. Azamimi, A., Uwate, Y., Nishio, Y.: An Improvement in Pattern Recognition Problem Using Chaotic BP Learning Algorithm. In: Proceedings of RISP International Workshop on Nonlinear Circuits and Signal Processing, pp. 213--216 (2009).
8. Krink, T., Paterlini, S.: Differential Evolution and Particle Swarm Optimization in Partitional Clustering. J. Computational Statistics and Data Analysis. 50, pp. 1220--1247 (2006).

9. Jarboui, B., Cheikh, M., Siarry, P., & Rebai, A.: Combinatorial particle swarm optimization (CPSO) for partitional clustering problem. J. Applied Mathematics and Computation. 192(2), pp. 337--345 (2007).
10. Frank, A. & Asuncion, A. (2010). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Hierarchical Traces for Reduced NSM Memory Requirements

Torbjørn S. Dahl

Abstract This paper presents work on using hierarchical long term memory to reduce the memory requirements of nearest sequence memory (NSM) learning, a previously published, instance-based reinforcement learning algorithm. A hierarchical memory representation reduces the memory requirements by allowing traces to share common sub-sequences. We present moderated mechanisms for estimating discounted future rewards and for dealing with hidden state using hierarchical memory. We also present an experimental analysis of how the sub-sequence length affects the memory compression achieved and show that the reduced memory requirements do not effect the speed of learning. Finally, we analyse and discuss the persistence of the sub-sequences independent of specific trace instances.

1 Introduction and Motivation

This paper presents a novel long term memory (LTM) structure that reduces the memory requirements of nearest sequence memory (NSM) learning. NSM is a simple instance-based reinforcement learning (RL) algorithm originally presented by McCallum [14]. While instance-based algorithms learn fast, one of their disadvantages is their relatively large memory requirements. The memory structure we present reduces the NSM memory requirements by allowing the *traces* [20] stored in LTM to share common sub-sequences.

Our aim with this work is to develop efficient RL algorithms that can scale up to complex learning problems such as robot control. We believe this can only happen when a number of properties have been brought together in a single algorithm. In terms of capability such an algorithm must be able to estimate future rewards accurately in the presence of hidden states. In terms of capacity it must, like the

Torbjørn S. Dahl
Cognitive Robotics Research Centre, University of Wales, Newport, Allt-yr-yn Avenue, Newport, NP20 5DA, United Kingdom, e-mail: torbjorn.dahl@newport.ac.uk

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_12,

human cortex, use both *hierarchy* and *auto-associativity* for efficient encoding [5]. In terms of structure and development they should, again like the human cortex, be *modular* and *constructivist* [17]. In terms of efficiency, they should, like humans and animals, be able to do *one-shot learning*, i.e., learning from from one or very few instances. For efficiency purposes, the algorithm should also be parallelisable. The work presented in this paper is a first step toward bringing these properties together in that it explores the principles of hidden state identification and discounted future reward estimation in a hierarchical, one-shot learning algorithm with the view to accommodating auto-associativity, modularity and constructivism in the future.

Section 2 of this paper places the work in the wider RL context. Section 3 presents the new hierarchical memory structure in detail along with the corresponding mechanisms for discounted future reward estimation and hidden state identification. Section 4 presents an experimental analysis of how the length of the subsequences affects the degree of re-use achieved as well as comparative experimental evidence indicating that the speed of the underlying NSM learning algorithm is not affected by the hierarchical memory representation. Section 5 presents an experimental analysis of the persistence of the shared sub-sequences. Finally, Section 6 concludes and indicates how we aim to extend this work in the future.

2 Related Work

Learning task hierarchies was identified by Dietterich [3] as an important challenge in RL and this was re-emphasised by Barto and Mahadevan [1]. Multiple frameworks for hierarchical RL have already been published, the most influential being the *options* formalism introduced by Sutton *et al.* [21]. Barto and Mahadevan [1] have presented a review of these and related frameworks. A lot of work has already been done on developing algorithms for hierarchical RL that automatically construct the abstractions used for the hierarchical representations. Digney [4] has presented the *Nested Q-learning* algorithm for this problem. Nested Q-learning uses environmental *features*, distinct recognisable sensory conditions to partition problem spaces. McGovern [15] has presented multiple algorithms for automated discovery of *sub-goals* based on bottle-necks defined by observation density in successful trajectories and *action sequences* based on frequency in successful trajectories. Hengst [6] has presented the *HEXQ* algorithm which uses the frequency of change in the state variables to define levels in the hierarchy. Sun and Sessions [19] have shown that, based on reward only, it is possible to construct dynamically and without prior knowledge of the problem domain, a hierarchy of individual agents who divide any given problem space in to beneficial sub problems and learn to solve these more efficiently than flat solutions. Common to these algorithms is their reliance on a statistical analysis of the input, something that requires a large number of trials in order to identify exploitable properties of the data. As such these algorithms are not *one-shot*.

Moore and Atkeson [16] have demonstrated that instance-based RL algorithms can improve on the speed of traditional algorithms such as temporal difference (TD) learning [20]. Other work in this direction deals with one-shot learning in terms of memory-based learning, e.g., locally weighted regression for robot juggling [18] and nonlinear oscillators for learning tennis strokes through programming by demonstration [8]. While these algorithms are one-shot, they are not hierarchical, and as such are unlikely to scale up to more general, less constrained problems.

Finally, at least two algorithms have integrated hierarchical representations with one-shot learning. McCallum [13] encoded trace instances hierarchically in the Utile Suffix Memory and U-Tree algorithms. These algorithms perform a much more efficient compression of memory than our algorithm in that is only distinguishes between states when there is a significant effect on the expected reward. However, there is no obvious way of implementing these algorithm using auto-associative structures and as such, these algorithms have little potential for further compression. Hernandez-Gardiol and Mahadevan [7] showed that hierarchical instance-based frameworks can improve the speed of learning beyond what is currently being achieved with flat representations. Hernandez-Gardiol and Mahadevan's work however, looked in particular at representing problems at different levels of granularity in pre-structured hierarchies. As a result, the algorithms are not as generally applicable as those that automatically construct hierarchical representations.

Our work approaches the problem of task hierarchies from the angle of reuse for compression and uses syntactically defined hierarchies primarily to reduce the memory requirements of NSM. The reduction in memory requirement is achieved by constructing, dynamically and without prior domain knowledge, a LTM where finite sequences of transitions are shared between traces. Reduced memory requirements is an important factor in scaling up RL algorithms to handle increasingly complex problems. In Section 6 we discuss how this work is the first step in our work on developing RL algorithms that are both constructivist and auto-associative.

3 NSM with Hierarchical Traces

Kaelbling *et al.* [9] have presented an RL formalism for problems with hidden state, i.e., partially observable Markov decision problems (POMDPs). Basic Markov decision problems (MDPs) are described as a four-tuple $< \mathscr{S}, \mathscr{A}, \mathscr{T}, \mathscr{R} >$ where $\mathscr{S}$ is a finite set of states of the world, $\mathscr{A}$ is a finite set of actions, $\mathscr{T}$ is the state transition function and $\mathscr{R}$ is the reward function. To describe POMDPs Kaelbling *et al.* extend the MDP framework to a six-tuple $< \mathscr{S}, \mathscr{A}, \mathscr{T}, \mathscr{R}, \Omega, \mathscr{O} >$ where Ω is a finite set of observations an agent can make of its world and $\mathscr{O}$ is the state observation function giving the probability of making a particular observation given a specific state and action. An OARO *transition* is a quadruple $< o, a, r, o' >$ where o is a starting observation, a is an action, r is the reward received and o' is the observation made in state s' resulting from taking action, a in the initial state s.

The original NSM learning algorithm was developed by McCallum [14] to be the simplest conceivable instance based RL algorithm. NSM learning is based on the k-nearest neighbours principle. It keeps track of the n latest observed OARO transitions in STM. In LTM it keeps the m most recently observed traces, where n and m are both fixed numbers. From the traces, NSM identifies the k OARO transitions that match most closely the last transitions stored in STM. The proximity measure used by NSM is the number of immediately preceding transitions in LTM that exactly match the immediately preceding transitions in STM. To select an action, NSM calculates the average discounted future reward (DFR) for each possible action over the k nearest neighbours. The NSM algorithm, in spite of being very simple, has been shown to compare favourably with other RL algorithms for problems with hidden state.

3.1 Hierarchical Traces

Figure 1 presents two different paths to a goal location in a grid-world originally presented by Sutton *et al.* [21].

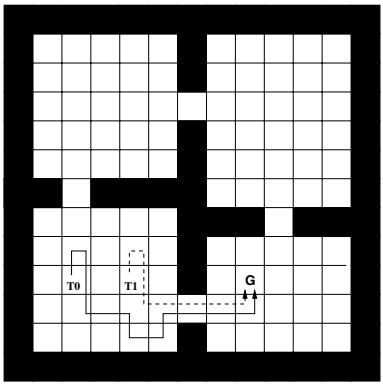


Fig. 1 Two different but overlapping paths to a goal location in the grid-world originally presented by Sutton *et al.* [21].

The values of the OARO transitions corresponding to the traces in Figure 1 are given in Table 1. Each row in Table 1 presents the values of a transition record as well as the corresponding DFR. We denote individual observations using the decimal representation of a four bits number where each bit indicates whether it's possible or not to enter the grid location to the north, east, south and west of the agent, respectively, starting from the least significant bit, e.g., a south-west corner with walls to the south and west would be represented bu the bit string 1100 and the decimal value 12. The available actions are movements in the same four directions,

iin the same order, represented as integers between 0 and 3 respectively. The reward function used yields a value of -1.0 when an agent attempts to move into a location which is occupied, a value of -0.1 when moving to an adjacent unoccupied location and a value of 5.0 when reaching the goal state. The DFR is calculated using a discount factor, γ, of 0.9.

Trace 0						Trace 1					
Record	o	a	r	o'	DFR	Record	o	a	r	o'	DFR
0	0	0	-0.1	0	0.88	12	0	0	-0.1	0	1.87
1	0	2	-0.1	0	1.09	13	0	2	-0.1	0	2.19
2	0	2	-0.1	0	1.32	14	0	2	-0.1	0	2.54
3	0	1	-0.1	0	1.58	15	0	1	-0.1	0	2.95
4	0	1	-0.1	0	1.87	8	0	1	-0.1	5	3.37
5	0	2	-0.1	4	2.19	9	5	1	-0.1	0	3.86
6	4	1	-0.1	6	2.54	10	0	1	-0.1	0	4.4
7	6	0	-0.1	0	2.95	11	0	0	5.0	0	5.0
8	0	1	-0.1	5	3.37						
9	5	1	-0.1	0	3.86						
10	0	1	-0.1	0	4.4						
11	0	0	5.0	0	5.0						

Table 1 The transition records of the two traces presented in Figure 1, including the discounted future rewards.

Each level of our hierarchical trace representation contains a set of fixed length sequences. The number of levels is only restricted by the size of STM. On the lowest level of the hierarchy, the elements in the sequences are references to single OARO transition records. On all the levels above, the sequence elements are references to sequences on the level below. For sequence size n, a sequence on level 0 will contain references to n transition records while a sequence on level m will contain references to n^{m+1} transition records. The hierarchical trace representation makes it possible for two traces with common sub-sequences to share a single record of that sub-sequence. Individual OARO transition records are also shared by multiple traces and common sub-sequences within a single trace can also share a single record.

The hierarchical trace representations are constructed whenever a goal location is reached. The hierarchical NSM algorithm (HNSM) then structures the trace recorded in STM into a hierarchy. Each of the sequences in this hierarchy is then compared to the sequences recorded in LTM and, if two sequences are identical, any references to the new copy in STM is replaced by references to the existing copy in LTM. Any new transition records or sequences are added at the appropriate level. Remaining transition records not included in the fixed size sequences are recorded in shorter sequences. When removing a trace from LTM, the component transition records and sequences are only removed if they are not referenced by other traces. The hierarchical representation of the traces representing the two paths in Figure 1 are presented graphically in Figure 2. The boxed numbers are references to the sequences or transition records on the level below. On level 0 they reference the OARO

transition records presented in Table 1. Note that both traces make use of sequence 2 on level 0. This is reflected in Table 1 by both traces having the same four last records.

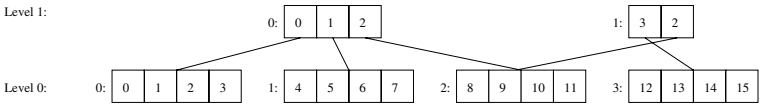


Fig. 2 A graphical representation of the traces representing the two paths presented in Figure 1 with sequence 2 on level 0 being shared by both traces.

3.2 Estimating Discounted Rewards

Calculating the DFR for each transition in a given trace is trivial with a linear representation. Here we present a mechanism for calculating DFR values using our hierarchical trace representation. In the hierarchical representation, all *elements* of a trace, i.e., both individual transition records and transition sequences, can be reused. It follows that each element can have multiple DFR values reflecting the different *contexts* in which they occur. We define a *local context* as a unique occurrence of an element in a superseding sequence. Note that an element can occur in multiple places in the same superseding sequence and also in different superseding sequences. Local contexts are indicated graphically by the lines connecting sequences and records across layers Figure 2. We define a *global context* as a unique occurrence of an element within a given trace.

Recording discounted rewards in the transition records means that two transition sequences are considered different if they occur at different distances from the final reward. This reduces the amount of calculation required at the cost of a lower level of memory reuse. We further reduced the memory requirements of our algorithm by recalculating the discounted rewards every time an action is chosen. We call this procedure *discounting rewards dynamically* (DRD). As DRD reliably reproduces the discounted rewards for all transitions in all the traces in LTM, we don't need to record rewards in the transition records. A given element can now be reused at any point in any trace.

An element that is used in multiple contexts correspondingly has multiple DFR values. The *local DFR* of a transition record or a sequence is the discounted sum of rewards recorded by that element. We denote the local DFR of memory element x in local context c, $d(x,c)$. The global DFR of a memory elements is the discounted sum of all the rewards recorded by preceding transition records in the given trace. In order to calculate the global DFR values of an element, the global DFRs of its superseding elements must be discounted over the number of transition records referred to by the elements preceding x in the given local context c. We denote this

distance $f(x,c)$. The discounted local DFRs for the preceding elements must then be added to the discounted global DFR of the superseding element. A recursive DFR definition is formalised in Equation 1. We use $D(x)$ to denote the set of global DFRs of an element and d' to denote a single global DFR for a superseding element. An element's set of local contexts is denoted $C(x)$. An element with no contexts has no superceding elements and thus, representent a complete trace.

$$D(x) = \begin{cases} \{0\} & \text{if } C(x) = \emptyset \\ \{d(x,c) + \gamma^{f(x,c)} d' \mid c \in C(x),\ d' \in D(c)\} & \text{otherwise} \end{cases} \quad (1)$$

Konidaris and Hayes [11] have provided evidence that DRD need not be computationally crippling. They have presented a mechanism called *asynchronous* RL within a greater behaviour-based RL framework. Asynchronous RL is a Q-learning mechanism that does *full backups* after each observed transition. A full backup improves a policy by iteratively refining it using estimated values to replace observations. This potentially requires revising the complete policy many times. Unlike asynchronous RL, DRD reproduces the same discounted rewards at each step and does not incrementally refine its estimates. What is important about Konidaris and Hayes' work in this context is their defence of the computationally expensive procedures of full backups on the basis that "...the time it takes a situated agent to move in the real world is very much longer than required to perform an update..." This argument also applies to DRD and Kondaris and Hayes' experimental evidence supports the feasibility of such approaches in general.

3.3 Hidden State Identification

In the same way that we developed a new mechanism for calculating DFR values for hierarchically represented traces, we have developed a corresponding mechanism for calculating the proximity values used to identify the k-nearest neighbours for handling hidden states. A *local proximity* is the number of matches between the transition records in STM and the transitions referred to by the preceding elements in a local context in LTM. A *global proximity* is the number of matches between STM and the transitions referred to by a global LTM context. However, only when STM matches all the preceding elements of a local context, is it necessary to check for further matches in higher level contexts. If there are further matches, the local proximity must be added to all of these to find the global proximities. A recursive definition of global proximity is formalised in Equation 2. The local proximity of an element given a context c is denoted $p(x,c)$ and the set of global proximities of an element $P(x)$. We also use $P(c)$ to denote the global proximity of a context, c. We use p' to denote the global proximity for a superseding element. Note that the global proximities are only added when STM matches the local context completely.

$$P(x) = \begin{cases} \{0\} & \text{if } C(x) = \emptyset \\ \{p(x,c) \mid c \in C(x)\} & \text{if c matches STM incompletely} \\ \{p(x,c) + p' \mid c \in C(x),\ p' \in P(c)\} & \text{if c matches STM completely} \end{cases} \quad (2)$$

We have used this definition of proximity to implement a function which calculates all possible proximities for each transition record. From these global proximities we choose the k highest and execute the action with the highest average DFR.

4 Re-use and Sub-Sequence Length

Our algorithm currently uses fixed-length sub-sequences, i.e., is syntactically defines the hierarchical structure of the LTM. However, the hierarchical hidden state identification and DRD mechanisms also support variable length sub-sequences. This makes it possible to vary the sub-sequence length to optimise memory compression or to consider semantic rather than syntactic criteria for sub-sequence lengths, e.g., the amount of re-use can be used to allow heavily referenced subsequences to grow longer, or the strength of reward signal can be usded to allow sub-sequences with high DFR values to be longer than sub-sequences with low DFR values. Such mechanisms provide a rich area for exploring and optimising memory usage. We discuss these opportunities further in Section 6. For fixed size subsequences, the optimal sub-trace size is not obvious and is likely to be problem specific. However, a sub-trace size of 1 would not provide any scope for compression, so the smallest possible sub-trace size is 2. Increasing the sub-trace size reduces the chance of matching sub-sequences. A sub-sequence size larger than the size of STM would mean that only complete traces would be shared. We have analysed the memory requirements by counting the number of values and references stored when using sub-sequences of size 2, 4 and 8 on two different problems described below. We have also analysed the memory usage for the original NSM algorithm.

For simplicity we use an abstract memory cost of 1 for observations, actions, resulting observations, rewards and DFR values as one each. A reference from a sequence to a sub-element or super-sequence also has a cost of 1. Each transition record in the original NSM algorithm thus has a memory requirement of 5 and a trace containing n transition records has a memory requirement of $5n$. Our algorithm dynamically discounts the rewards, thus the memory requirements of each transition record is 4 as we do not store the DFR value. However, in addition to the transition records, our algorithm needs two-way references between elements on different levels in the hierarchy in order to support hierarchical hidden state identification and DRD. While each transition records has a memory requirement of $4n$, there is also the overhead of hierarchical references. A complete hierarchical trace without reuse contains $2(n \log n)$ references and has a total memory requirement of $4n + 2(n \log n)$. The actual number of transition records and references used by our algorithm will be smaller than this as different traces will share sub-sequences. We

ran our algorithm in two different abstract mazes, or *grid worlds* [1] , taken from the RL literature. The first grid-world was originally published by Sutton [21] and is presented graphically in Figure 1. The second was originally published by by McCallum [14] and is presented graphically in Figure 3.

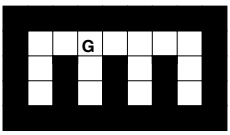


Fig. 3 The simplest of the two grid worlds used for the experiments, originally presented by McCallum [14].

All the experiments used the reward function and discount factor given in Section 3.1. The number of nearest neighbours, k, was set to 7, and the exploration rate was fixed at 0.1. The size of the STM, i.e., the maximum trace length, was 40 and the size of the LTM, i.e., the maximum number of traces stored, was also 40. Each experiment consisted of 40 trials and each trial consisted of 200 runs. The average memory usage for McCallum's world is presented graphically in Figure 4. The usage for Sutton's world is presented in Figure 5. The different versions of our algorithm, using DRD and hierarchical traces is labelled *DHNSMn*, where n denotes the sub-sequence length. The average memory usage for the original NSM algorithm is also given. The x-axis gives the number of runs the algorithms have gone through while the y-axis gives the average memory usage.

The results presented in Figures 4 and 5 show that in both grid-worlds, the lowest memory requirements are achieved when using a sub-trace size of 4. We also performed experiments using sub-trace sizes of 3, 5, 6 and 16. Sub-sequences of 3 and 5 produce a memory usage that is not significantly different from those produced by sub-sequences of size 4. Sub-sequences of size 6 and 16, like 8, produce a memory usage that is significantly higher. For the experiments on learning speed and sub-sequence persistence below, we only consider sub-sequences of size 4.

4.1 Speed of Learning

The original NSM algorithm and our algorithm are effectively the same algorithm with different memory requirements. As a point of interest, when debugging our algorithm, we ran the two in parallel and compared the proximities, DFRs and actions selected in order to ensure that they were exactly the same. As a result, the perfor-

[1] All our experiments were conducted using the Rumpus open source, stand-alone grid world server which supports easy development of new RL problems using images and XML as well as multi-language support through TCP sockets. The Rumpus server along with all its implementations of RL problems and algorithms are available under the GNU general licence from http://rumpus.rubyforge.org.

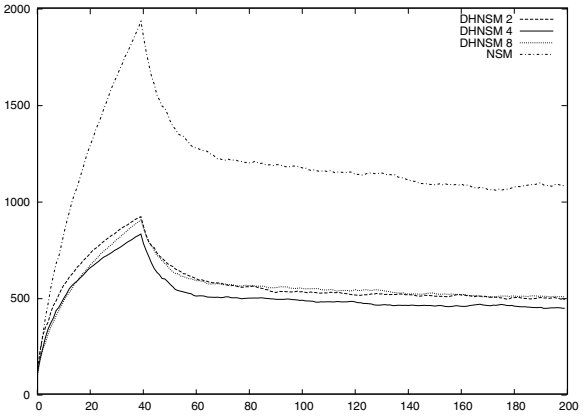


Fig. 4 The memory usage of our DHNSM algorithm using different sub-sequence lengths in McCallum's grid world.

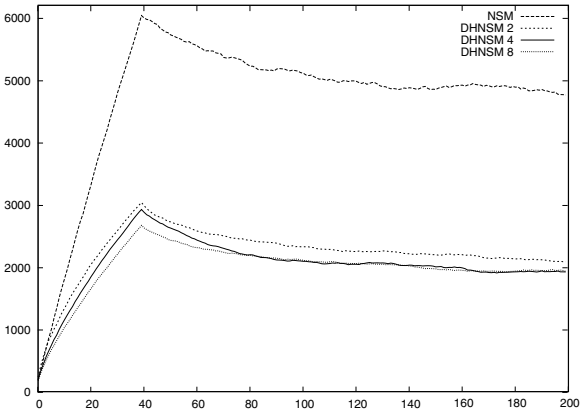


Fig. 5 The memory usage of our DHNSM algorithm using different sub-sequence lengths in Sutton's world

mance is in time should be indistinguishable. In Figure 6 we present graphically the performance in time from the experiments in McCallum's grid-world described above. In Figure 7 we present the same data for Sutton's world. Again the x-axis gives the number of runs the algorithms have gone through while the y-axis gives the average number of steps needed to reach the goal.

The data presented in Figures 7 and 6 indicate that there is no significant difference in terms of the speed of learning, between the original NSM algorithm and our algorithm independent of sub-sequence length.

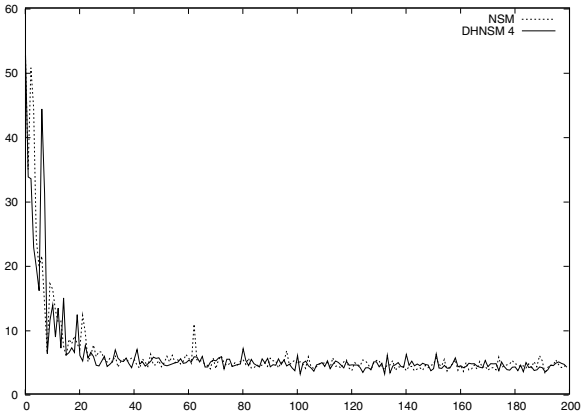


Fig. 6 The average number of steps needed to complete each run for the DHNSM 4 and NSM algorithms in McCallum's world

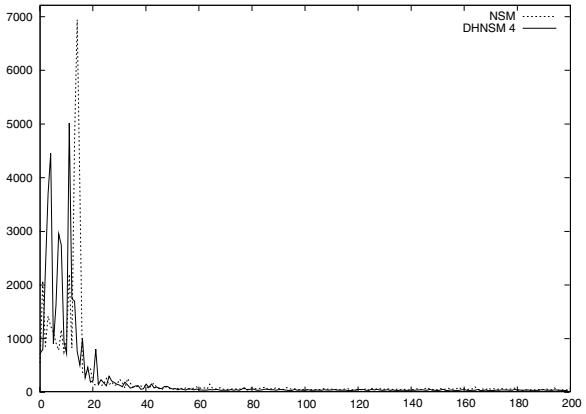


Fig. 7 The average number of steps needed to complete each run for the DHNSM 4 and NSM algorithms in Sutton's world

5 Memory Persistence

As hierarchical traces are made up of shareable sub-sequences, there is a potential for sub-sequences to out-live the traces that originally created them. Sub-sequences that persist beyond the lifetime of any individual trace describe commonly reccurring agent-world interaction sequences. Sub-sequences that persist throughout the learning process describe interactions that are both common and beneficial in that they are part of the successful traces that are repeated more frequently as the agents learns. In Figure 8 we present the maximum, average and minimum age of the sub-

sequences from McCallum's world. Figure 9 presents the same data from Sutton's world.

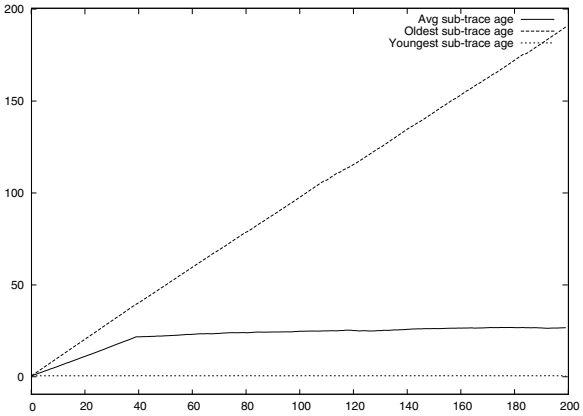


Fig. 8 The maximum, average and minimum age of the sub-sequences in LTM for the DHNSM 4 algorithm in McCallum's world

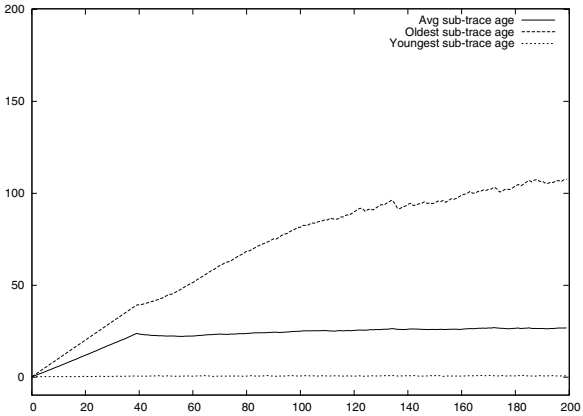


Fig. 9 The maximum, average and minimum age of the sub-sequences in LTM for the DHNSM 4 algorithm in Sutton's world

Figures 9 and 8 both show the presence of a persistent core of sub-sequences. If sub-sequences did not outlive their creators, the average sub-trace length would be 20. For both worlds, the average grows to over 25 with the maximum age growing to over 100. These persistent elements describe the agent-world interaction by representing its most common transition sequences. As such these elements form a bridge between instance-based and model-based learning algorithms. The presence

of persistent elements of sub-sequences is also a promising sign for our future development of this algorithm in a more obviously model-based direction as described in Section 6.

6 Conclusions and Future Work

The results presented in this paper show that is is possible to use hierarchical traces for instance-based RL and that such traces can reduce the memory requirements of instance-based RL algorithms. While variable sub-sequence sizes are an interesting direction in which to take this research, another direction of enquiry provides a more compelling memory compression mechanism. We plan to implement sub-sequences as self-organising maps (SOMs) [10] and by doing this, we hope to achieve three things. First, we hope to provide further memory compression by representing similar traces through their principal components. Second, we hope to provide RL algorithms that are at once, sequential, hierarchical and auto-associative. Third, by implementing DFR and hidden state identification in terms of *activation spreading* [12] in a hierarchical SOM, we aim to produce RL algorithmjs that are massively parallelisable on a low, non-von Neumann architectures such as field programmable gate arrays. Chang *et al.* have already demonstrated such paralellisation of traditional SOMs [2].

Such algorithms will have increased scalability in space and time. The work presented here gives us a formal yard-stick for evaluating the performance of such a mechanism.

References

1. Barto, A.G., Mahadevan, S.: Recent advances in hierarchical reinforcement learning. Dicrete Event Dynamic Systems: Theory and Applications **13**, 41–77 (2003)
2. Chang, C.H., Shibu, M., Xiao, R.: Salf organizing feature map for color quantization on FPGA. In: A.R. Omondi, J.C. Rajapakse (eds.) FPGA Implementations of Neural Networks. Springer (2006)
3. Dietterich, T.G.: Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Artificial Intelligence Research **13**, 227–303 (2000)
4. Digney, B.L.: Emergent hierarchical control structures: Learning reactive/hierarchical relationships in reinforcement learning. In: P. Maes, M. Matarić, J.A. Meyer, J. Pollack, S.W. Wilson (eds.) From Animals to Animats 4 [Proceedings of the 4th International Conference on Simulation of Adaptive Behavior (SAB'06), Cape Cod, Massachusetts, September 9 - 13, 1996], pp. 363–372. MIT Press/Bradford Books (1996)
5. Fuster, J.M.: Cortex and Mind: Unifying Cognition. Oxford University Press, New York (2003)
6. Hengst, B.: Discovering hierarchy in reinforcement learning with HEXQ. In: C. Sammut, A.G. Hoffmann (eds.) Machine Learning [Proceedings of the 19th International Conference (ICML'02), Sydney, Australia, July 8 - 12, 2002], pp. 243–250. Morgan Kaufmann (2002)
7. Hernandez-Gardiol, N., Mahadevan, S.: Hierarchical memory-based reinforcement learning. In: T.K. Leen, T.G. Dietterich, V. Tresp (eds.) Advances in Neural Information Processing Sys-

tems 13 [Proceedings of the NIPS Conference, Denver, Colorado, November 28 - 30, 2000], pp. 1047–1053. MIT Press (2001)
8. Ijspeert, A.J., Nakanishi, J., Schaal, S.: Movement imitation with nonlinear dynamical systems in humanoid robots. In: Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA'02), pp. 1398–1403. Washington, DC (2002)
9. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence **101** (1998)
10. Kohonen, T.: Self-organizing maps, 3rd edn. Springer, New York (2001)
11. Konidaris, G.D., Hayes, G.M.: An architechture for behavior-based reinforcement learning. Adaptive Behavior **13**(1), 5–32 (2005)
12. Maes, P.: How to do the right thing. Connection Science **1**(3), 291–232 (1989)
13. McCallum, A.: Instance-based utile distinctions for reinforcement learning with hidden state. In: Proceedings of the 12th International Conference on Machine Learning (ICML'95), pp. 387–395. Tahoe City, California (1995)
14. McCallum, A.: Hidden state and reinforcement learning with instance-based state identification. IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics (Special Issue on Robot Learning) **26**(3), 464–473 (1996)
15. McGovern, A.: Autonomous discovery of temporal abstractions from interactions with an environment. Ph.D. thesis, University of Massachusetts, Amherst, Amherst, Massachusetts (2002)
16. Moore, A.W., Atkeson, C.G.: Prioritized sweeping: Reinforcement learning with less data and less time. Machine Learning **13**, 103–130 (1993)
17. Quartz, S.R.: Learning and brain development: A neural constructivist perspective. In: P.T. Quinlan (ed.) Connectionist Models of Development: Developmental Processes in Real and Artificial Neural Networks, pp. 279–310. Psychology Press (2003)
18. Schaal, S., Atkeson, C.G.: Robot juggling: Implementation of memory-based learning. Control Systems Magazine **14**(1), 57–71 (1994)
19. Sun, R., Sessions, C.: Self-segmentation of sequences: Automatic formation of hierarchies of sequential behaviors. IEEE Transactions on Systems, Man and Cybernetics: Part B, Cybernetics **30**(3), 403–418 (2000)
20. Sutton, R.S., Barto, A.G.: Reinforcement learning: an introduction. MIT Press, Cambridge, Massachusetts (1998)
21. Sutton, R.S., Precup, D., Singh, S.P.: Between MDPs ans semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence **112**, 181–211 (1999)

On Reinforcement Memory for Non-Markovian Control

Hassab Elgawi Osman

Abstract This paper contributes on designing robotic memory controller for solving non-Markovian reinforcement tasks, which correspond to a great deal of real-life stochastic predictions and control problems. Instead of holistic search for the whole memory contents, the controller adopts associated feature analysis to produce the most likely relevant action from previous experiences. Actor-Critic (AC) learning is used to adaptively tune the control parameters, while an on-line variant of decision-trees ensemble learner is used as memory-capable to approximate the policy of the Actor and the value function of the Critic. Learning capability is experimentally examined through non-Markovian cart-pole balancing task. The result shows that the proposed controller acquired complex behaviors such as balancing two poles simultaneously.

1 Introduction

Neuroscientists believe that living beings solve the daily life activities, making decisions and hence adapt to newly situations by learning from past experiences. Learning from experience implies that each event is learnt through features (i.e. sensory control inputs) analysis, aimed at specifying and then recalls more important features for each event or situation.

In robot learning, several works seem to suggest that the transition to the current reinforcement learning (RL) [1], as a general formalism, does correspond to observable mammal brain functionality, where 'basal ganglia' can be modeled by an Actor-Critic (AC) version of *temporal difference* (TD) learning [2, 3, 4]. However, as with the most real-world learning systems, the arising of 'perceptual aliasing' [5] (also referred to as a problem of 'incomplete perception', or 'hidden state'), when the system has to scale up to deal with complex non-linear search spaces in a non-

Hassab Elgawi Osman
The University of Tokyo, Tokyo-Japan

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_13,

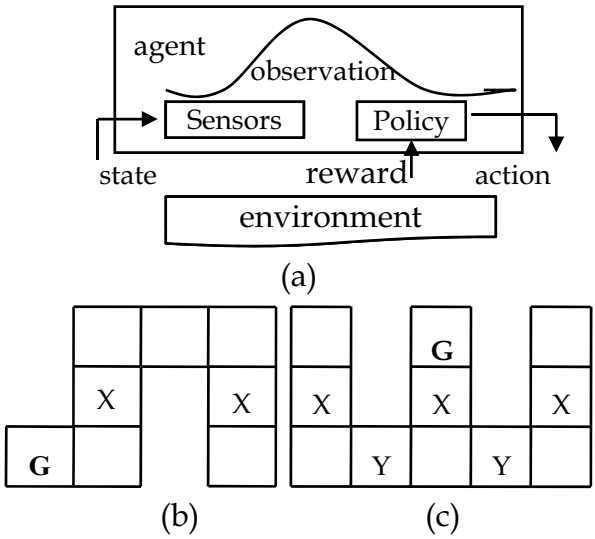


Fig. 1 POMDP and Perceptual aliasing. RL agent is connected to its world via perception state S and action A. In (a) a partially observable world, in which the agent does not know which state it is in due to sensor limitations; for the value function v^{π}, the agent updates its policy parameters directly. In (b) and (c) two maze domains. States indicated with the same letter (X or Y) are perceptually aliased because the agent is sensed only wall configuration.

Markov settings or *Partially Observation Markov Decision Process* (POMDP) domains [6] (see Fig. 1) renders to-date RL methods impracticable, and that they must learn to estimate *value function* v^{π} instead of learning the *policy* π, limiting them mostly for solving only simple learning tasks, raising an interest in heuristic methods that directly and adaptively modifying the learning policy $\pi : S \rightarrow A$, which maps perceptual state/observation S to action A via interaction with the rest of the system [7, 8].

Inclusion of a memory to a simulated robot control system is striking because a memory learning system has the advantage to deal with perceptual aliasing in POMDP, where memoryless policies are often fail to converge [9].

In this paper, a self-optimizing memory controller is designed particularly for solving non-Markovian tasks, which correspond to a great deal of real-life stochastic predictions and control problems (Fig. 2). Rather than holistic search for the whole memory contents the controller adopts associated feature analysis to successively memorize a newly experience (state-action pair) as an action of past experience [10]. e.g., If each past experience was a chunk, the controller finds the best chunk for the current situation for policy exploration. Our aim is not to mimic the neuroanatomical structure of the brain system but to catch its properties, avoids manual 'hard coding' of behaviors. AC learning is used to adaptively tune the control parameters, while an on-line variant of decision-tree ensemble learner [11, 12] is used as memory-capable function approximator to approximate the policy of the *actor* and

the value function of the *critic*.
Section 2 briefly highlights on POMDP settings. A description with comprehensive illustration of the proposed memory controller will be given in Section 3. Then Section 4 highlights a comparison of conventional memory controller and the self-optimizing memory controller. Section 5 shows the implementation of decision-tree ensemble as memory-capable function approximator for both critic and policy. Some experimental results are presented in Section 6 as promising examples. It includes the non-Markovian cart-pole balancing tasks. The results show that our controller is able to memorize complete non-Markovian sequential tasks and develop complex behaviors such as balancing two poles simultaneously.

2 A non-Markovian and Perceptual Aliasing

First we present the formal setting of POMDP and then highlight on related approaches tacking perceptual aliasing.

2.1 POMDP Formal Setting

The formal setting of POMDP is $\mathscr{P} = \langle \mathscr{M}, \mathscr{O}, \mathscr{Z} \rangle$ consist of:

1. An MDP of a tuple $\mathscr{M} = \langle S, A, T, R \rangle$ where S is the space of possible *states* of the environment, A is a set of *actions* available to the agent (or control input), $P : S \times A \times S \to [0,1]$ defines a conditional probability distribution over *state transitions* given an action, and $R : S \times A \to R$ is a *reward function* (payoff) assigning a reward for an action,
2. A set of possible observations $\mathscr{O}$, where $\mathscr{O}$ could constitute either a set of discrete observations or a set of real-value,
3. $\mathscr{Z}$, a probability density mapping state-observation combinations $S \times \mathscr{O}$ to a probability distribution, or in the case of discrete observations combinations $S \times \mathscr{O}$ to probabilities. In other words, $Z(s,o)$ yields the probability to observing o in state s. So basically, a POMDP is like an MDP but with observations instead of direct state perception.

If a world model is made available to the controller, it can easily calculate and update a *belief vector* $\mathbf{b_t} = \langle b_t(s_1), b_t(s_2), \cdots, b_t(s_N) \rangle$ over 'hidden states' at every time step t by taking into a account the history trace $h = o_1, o_2, \cdots, o_{t-1}, o_t$.

3 Self-Optimizing Controller Architecture

One departing approach from manual 'hard coding' of behaviors is to let the controller build its own internal 'behavior model'–'on-the-fly' by learning from past experience. Fig. 2 illustrates the general view of our memory controller based on heuristic memory approach. We briefly explain its components. It is worth noted that in our implementation only the the capacity of the memory and reward function have be specified by a designer, the controller is self-optimized in a sense that we do not analyzing a domain *a priori*, instead we add an initially suboptimal model, which is optimized through learning[1].

Past experiences. Sensory control inputs from environment would be stored at the next available empty memory location (chunk), or randomly at several empty locations.

Feature predictor. Is utilized to produce associated features for each selective experience. This predictor was designed to predict multiple experiences in different situations. When the selective experience is predicted, the associated features are converted to feature vector so the controller can handle it.

Features Map. The past experiences are mapped into multidimensional feature space using neighborhood component analysis (NCA) [14, 15], based on the Bellman error, or on the temporal difference (TD) error. In general this is done by choosing a set of features which approximate the states S of the system. A function approximator (FA) must map these features into V^{π} for each state in the system. This generalizes learning over similar states and more likely to increase learning speed, but potentially introduces generalization error as the feature will not represent the state space exactly.

Memory access. The memory access scheduling is formulated as a RL agent whose goal is to learn automatically an optimal memory scheduling policy via interaction with the rest of the system. A similar architecture that exploits heterogeneous learning modules simultaneously has been proposed [16]. As can be seen in the middle of Fig. 2 two scenarios are considered. In (a) all the system parameters are *fully observable*, the agent can estimate v^{π} for each state and use its actions (e.g., past experiences). The agent's behavior, B, takes actions that tend to increase the long-run sum of values of the *reinforcement signal*, typically $[0,1]$. In (b) the system is *partially observable* as described in Fig. 1. Since our system is modeled as POMDP decision depends on last observation-action, and the observation transitions $s_{t+1} = \delta(s_t, a_t)$ depend on randomly past perceptual state. This transition is expressed by $Pr(s_t|s_{t-1}, a_{t-1}, s'_t, s''_t, \cdots)$, where s_{t-1}, a_{t-1} are the previous state and

[1] At this point we would like to mention that M3 Computer Architecture Group at Cornell has proposed a similar work [13] to our current interest. They implement a RL-based memory controller with a different underlying RL implementation, we inspired by them in some parts.

action, and t', t'' are arbitrary past time.

Learning behaviors from past experience. On each time step t, an *adaptive critic* (that is a component of the TD learning), is used to estimate future values of the *reinforcement signal* of retaining different memory locations, which represents the agent's behavior, B in choosing actions. The combinations of memory locations show to have the highest accumulated signals are more likely to be remembered. TD error–the change in expected future signal is computed based on the amount of occasional intrinsic reinforcement signal received, a long with the estimates of the adaptive critic.

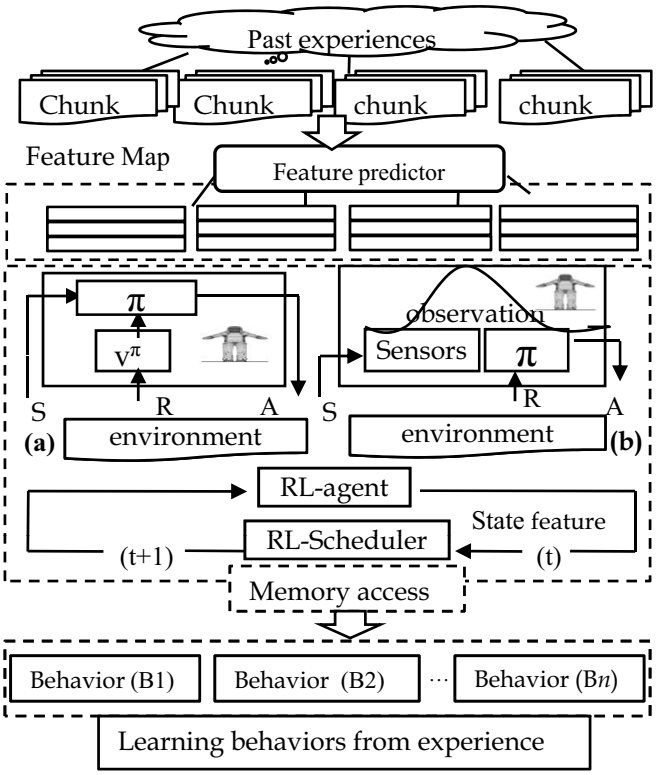


Fig. 2 Architecture of self-optimizing memory controller. The controller utilizes associated feature analysis to memorize complete non-Markovian reinforcement task as an action of past experience. The controller can acquired behaviors such as controlling objects, displays long-term planning and generalization capacity.

4 Non-Markovian Memory Controller

4.1 Conventional Memory Controller

Conventional manually designed memory controller suffers two major limitations in regard with scheduling process and generalization capacity. First, it can not anticipate the long-term planning of its scheduling decisions. Second, it lacks learning ability, as it can not generalize and use the experience obtained through scheduling decisions made in the past to act successfully in new system states. This rigidity and lack of adaptivity can lead to severe performance degradation in many applications, raising interest in self-optimizing memory controller with generalization capacity.

4.2 Self-Optimizing Memory Controller

The proposed self-optimizing memory controller is a fully-parallel maximum-likelihood search engine for recalling the most relevant features in the memory of past. The memory controller considers the long-term planning of each available action. Unlike conventional memory controllers, self-optimizing memory controller has the following capabilities: 1) Utilizes experience learnt in previous system states to make good scheduling decisions in new, previously unobserved states, 2) Adapts to the time-variant system in which the state transition function (or probability) is permitted to gradually change through time, and 3) Anticipates the long-term consequences of its scheduling decisions, and continuously optimizes its scheduling policy based on this anticipation.
No key words or pre-determined specified memory locations would be given for the stored experiences. Rather a parallel search for the memory contents would take place to recall the previously stored experience that correlates with the current newly experience. The controller handle the following tasks: (1) relate states and actions with the occasional reward for long planning, (2) take the action that is estimated to provide the highest reward value at a given state, and (3) continuously update long-term reward values associated with state-action pairs.

5 Memory-Capable Function Approximation

5.1 Actor-Critic Learning

Actor-critic (AC), a group of on-policy TD methods, separates the π and the v^{π} into independent memory structures. The π structure, or *actor*, is used to decide which action to pick in each state. The estimate of v^{π}, or *adaptive critic*, determines whether the actions of the *actor* are to be rewarded or punished. The algorithms use

these spare measures of performance to adopt an optimal behavior over time. The adaptive critic maps its current state event onto an estimate of whether it will be rewarded. The mapping is learned from the past experience. If $s+1$ is the situation that follows situation s in time, this expected future reward may be written as:

$$V(s) = \gamma^0 r(s) + \gamma^1 V(s+1) + \cdots + \gamma^n V(s+n) \tag{1}$$

The value of the current situation, $V(s)$, is the sum of all the rewards we will receive over the next n time steps. The rewards on each time step are "discounted" by factor, γ, in the range $[0,1]$. Equation (1) can be rewritten in a recursive form:

$$V(s) = \gamma^0 r(s) + \gamma^1 V(s+1) = r(s) + \gamma V(s+1) \tag{2}$$

It should be noted that the equality in Eq. (2) is valid only if n is infinite or the state at n time steps later, $s+n$, is always a so-called 'absorbing state.' Obviously a value function estimates that fall far from this equality in considered inaccurate, and the error is estimated based on TD error:

$$\delta(s) = (r(s) + \gamma V(s+1) - V(s)) \tag{3}$$

Adopting these methods can save much computation for selecting optimal actions, due to utilizing separate memory for value function and policy.

5.2 AC in non-Markovian Domain

Due to non-Markovian characteristics, the controller infers the state of its environment from a sequence of observations it receives, learns an optimal action by detecting certain past events, that associated with its current perception. In particular, at time t, the error of the critic is given by,

$$E_c(t) = \frac{1}{2}([r(t) + \gamma J(t)] - J(t-1))^2 \tag{4}$$

while the error of the actor is

$$E_a(t) = \frac{1}{2}(J(t) - R^*)^2 \tag{5}$$

where R^* is the optimal return, which is dependent on the problem definition. The expected return is expressed as the general utility function, $J(t)$, which is to be maximized by the controller. Specifically,

$$J(t) = r(t+1) + \gamma r(t+2) + \gamma^2 r(t+3) + \cdots \tag{6}$$

where $r(t)$ is the immediate reward and γ is the time-discounting factor $0 \leq \gamma \leq 1$.

5.3 Decision-tree Ensemble Memory for Optimal Learning

On-line decision-tree ensemble learner has the characteristics of a simple structure, strong global approximation ability and a quick and easy training [11, 12]. It has been used with TD learning for building a hybrid function approximator [18, 19]. Here, in order to improve learning efficiency and to reduce the demand of storage space and to improve learning efficiency, the on-line ecision-tree ensemble approximator is structured in a way that both *actor* and *critic* can be embodied in one structure, subsequently, is used to approximate π of the *actor* and the v^{π} of the *critic* simultaneously. That is, the actor and the critic can share the input and the basis functions structure of the decision tree. Let DT_{Appro} represents a hybrid approximator that combines actor and critic. Given a state $s(t)$ and action $a(t)$, DT_{Appro} is defined such that $DT_{Appro}(s(t),a(t)) = (J(t),a(t+1))$, where $J(t)$ is the estimated value of the given state-action pair, and $a(t+1)$ is the subsequent action to be taken by the controller. At the critic output the error is captured by TD error. However, at the action outputs the error is determined by the gradient of the estimated value $J(t+1)$ w.r.t the action $a(t+1)$ selected by the on-line RF at time t. Specifically,

$$e_a(t) = \alpha \nabla_{a(t+1)} J(t+1)$$

$$= \alpha \left(\frac{\partial J(t+1)}{\partial a_1(t+1)}, \cdots, \frac{\partial J(t+1)}{\partial a_d(t+1)} \right) \tag{7}$$

where α is a scaling constant and d is the choices availabilities at action a. Accumulating the error for each choice of the selected action, the overall actor error is given be:

$$E_a(t) = \frac{1}{2} \left[\sum_{i=1}^{d} e_{ai}^2(t) \right] \tag{8}$$

where $e_{ai}(t)$ is the choice of the action error gradient $e_a(t)$. In finding the gradient of the estimated value $J(t+1)$ w.r.t the previously selected action $a(t+1)$, the direction of change in action, which will improve the expected return at time step $t+1$, is obtained. Thus by incrementally improving actions in this manner, an optimal policy can be achieved. $E(t) = E_c(t) + E_a(t)$ defines the reduced error for the entire on-line appriximator.

6 Experiment and Results

As discussed in previous sections, the proposed controller brings a number of preferable properties for learning different behaviors. In this section, we investigate its learning capability through a task of cart-pole balancing problem, designed with non-Markovian settings.

6.1 Related work

Modeling the pole balancing algorithm for POMDP has received much interest in the field on control and artificial intelligence. Although a variation of Value and Policy Search (VAPS) algorithm [20] has been applied to this problem for the POMDP case [21], they have assumed that the position of cart on track x and the angle of pole from vertical θ are completely observable. NeuroEvolution of Augmenting Topologies [22] and evolutionary computation [23], are another promising approaches where recurrent neural networks are used to solve a harder balancing of two poles of different lengths, in both Markovian and non-Markovian settings.

6.2 Non-Markovian Cart Pole Balancing

As illustrated in Fig. 3A, Cart-Pole balancing involves a vertical pole with a point-mass at its upper end installed on a cart, with the goal of balancing the pole when the cart moves by applying horizontal forces to the cart, which must not stray too far from its initial position. The state description for the controller consists of four continuous state variables, the angle θ (radial), and the speed of the pole $\dot{\theta} = \delta x/\delta t$ plus the position x and speed of the cart $\acute{x} = \delta x/\delta t$, (see Appendix A for the equations of motion and parameters used as reported by [23]). The two continuous actions set up for controller training and evaluation were RightForce (RF), (results in pushing the cart to the right), and LeftForce (LF), (results in pushing the cart left). At each time step t, the controller must only observe the θ (that is, the controller is not observing the velocities $(\dot{x}, \dot{\theta})$) and then takes appropriate action to balance the pole by learning from the past experience and the intrinsically rewards. The optimal value function is shown in Fig. 3B. A simulated sample run is shown in Fig. 4. The controller could keep the pole balanced after about 4000 steps.

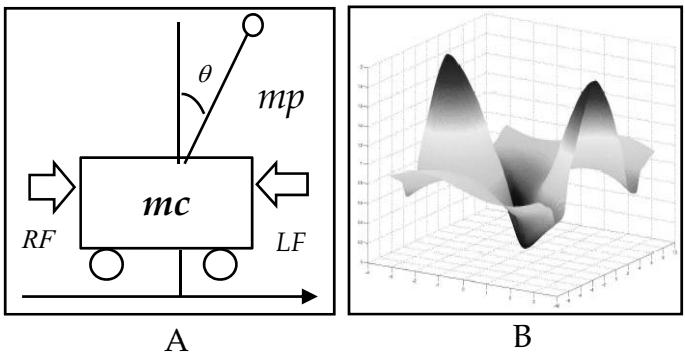


Fig. 3 (A) Illustration of the non-Markov Cart-Pole balancing problem, where the angular velocity is not observing by the controller. (B) Optimal value function.

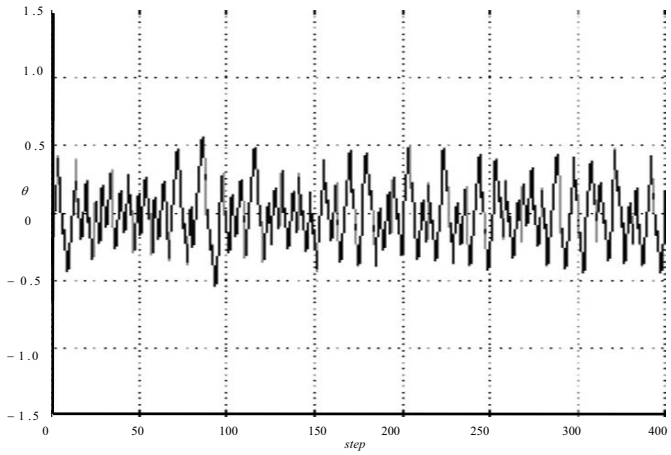


Fig. 4 A sample learning for balancing the pole. It suggests that the method could keep the pole near the top for a long time.

6.3 Non-Markovian Two-Pole Balancing

Then we moved to a harder setting of this problem, balancing two poles simultaneously (see Fig. 5). Each pole has its own position and angular velocity, θ_1 and $\dot{\theta}_1$ for the first pole and θ_2 and $\dot{\theta}_2$ for the second pole respectively. The controller must balance the two poles without velocity information. In order to assist the feasibility of our approach to balance two poles simultaneously we compared with other methods.

Table 1 reports the performance of our controller compared with traditional value

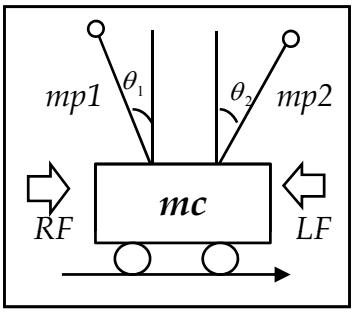


Fig. 5 Illustration of the non-Markov 2-Pole balancing problem. Parameters known are θ_1 and θ_2. The controller must balance the two poles without observing $\dot{\theta}_1$ and $\dot{\theta}_1$.

function-based methods (including SARSA-CMAC, SARSA-CABA, which are reported by [23], who used SARSA implementations by [24] and VAPS) and policy search method (including Q-MLP, as implementation of [23]). The parameter settings of each methods is reported in Appendix B. Table 1 shows that our controller takes the minimal evaluations to balance the poles. With regard to CPU time (reported in seconds) we slightly fall short to Q-MLP. However, it interesting to observe that none of the value function approaches could handle this task in within the set of steps (e.g., 100,000 time steps, which is equal to over 30 minutes in simulated time) due to the memory constraint. The result also indicates that our memory controller stand as a promising method in solving this benchmark more successful than the traditional RL techniques.

Table 1 Comparison of our result for balancing two-pole simultaneously with other value function approaches and policy based methods. 'Evaluation' indicates the total time steps for the method to be able to keep the poles near the top for a long time.

	Method	Evaluation	time (second)
V-function	SARSA-CMAC	Time Out	-
	SARSA-CABA	Time Out	-
	VAPS	Time Out	-
Policy	Q-MLP	10,582	153
Memory	Our	8,900	300

7 Conclusions

This paper proposes an architecture which avoids manual 'hard coding' of behaviors, where an RL agent uses an adaptive memory process to create its own memory and thereby perform better in partially observable domains. The algorithm uses neighborhood component analysis (NCA) to determine feature vectors for system states. Decision-trees ensemble is used to create features which are useful in predicting the state of the system (i.e. building some sort of forward model). Chunks are used with a feature predictor to get features. These features are then used as the input features to learn a policy. Results based on non-Markov Cart-Pole balancing indicate that our model can memorize complete non-Markovian sequential tasks and is able to produce behaviors that make the controlled system to behave desirably in the future. One of our future plans is to automate the capacity of memory in order to accommodate more complex tasks. In our current design the number of chunks that can be used is fixed. Another future plan will be in designing intelligent mechanism for memory updating, and to experiment with real world applications.

References

1. Sutton, R., Barto, A.: "Reinforcement Learning: An introduction,". *Cambring, MA: MIT Press* (1998).
2. Barto A.: "Adaptive critics and the basal ganglia,". In: *Models of Information Processing in the Basal Ganglia*, pp.215-232. Cambridge, MA: MIT Press (1995).
3. Suri, R., Schultz, W.: "A neural network model with dopamine-like reinforcement signal that learns a spatial delayed response task,". In: *Neuroscience* **91**(3):871-890 (1999).
4. Suri, R., Schultz, W.: "Temporal difference model reproduces anticipatory neural activity,". In: *Neural Computation* **13**:841-862 (2001).
5. Chrisman, L.: "Reinforcement learning with perceptual aliasing: The perceptual distinctions approach,". In: *Proc. Int'l. Conf on AAAI*, pp.183-188 (1992).
6. Cassandra, A., Kaelbling, L., Littman, M.: "Acting optimally in partially observable stochastic domains,". In: *Proc. Int'l. Conf on AAAI*, pp.1023-1028 (1994).
7. Sutton, R., McAllester, D., Singh, S., Mansour, Y.: "Policy gradient methods for reinforcement learning with function approximation,". In: *Advances in Neural Information Processing Systems* 12, pp. 1057-1063. MIT Press (2000).
8. Aberdeen, D., Baxter, J.: "Scalable Internal-State Policy-Gradient Methods for POMDPs,". In: Proc. of *19th Int'l Conf. on Machine Learning* **12**, pp.3-10. Morgan Kaufmann Publishers Inc. (2002).
9. Tsitsiklis, J., Van Roy, B.: "Featured-based methods for large scale dynamic programming,". In: *Machine Learning* **22**:59-94 (1996).
10. Hassab Elgawi, O.: "RL-Based Memory Controller for Scalable Autonomous Systems," In: *Advances in Neuro-Information Processing*, Chi-Sing Leung, Minho Lee, Jonathan Hoyin Chan (Eds.), Part II, LNCS 5864, pp.83-92, (2009).
11. Basak, J.: "Online adaptive decision trees: Pattern classification and function approximation,". *Neural Comput* **18**:2062-2101 (2004).
12. Hassab Elgawi, O.: "Online Random Forests based on CorrFS and CorrBE," In In: Proc. of *Conf on Computer Vision and Pattern Recognition Workshop*, CVPR, pp.1-7 (2008).
13. Ipek, E., Mutlu, O., Martinez, J., Caruana, R.: "Self-Optimizing Memory Controllers: A Reinforcement Learning Approach,". In: *Intl. Symp. on Computer Architecture (ISCA)*, pp.39-50 (2008).
14. Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: "Neighbourhood Components Analysis,". In: *Advances in Neural Information Processing Systems* **17**, MIT Press, pp.513-520 (2005).
15. Keller, P., Mannor, S., Precup, D.: "Automatic basis function construction for approximate dynamic programming and reinforcement learning,". In: *23rd International Conference on Machine Learning*, pp.449-456 (2006).
16. Uchibe, E., Doya, K.: (2006) "Competitive-Cooperative-Concurrent Reinforcement Learning with Importance Sampling,". In: Proc. of the *Eighth Int'l Conf. on Simulation of Adaptive Behavior: From Animals to Animats*, 8, MIT Press,Cambridge, MA, 2004, pp.287-296.
17. Leslie P., Michael L., Anthony R. "Planning and acting in partially observable stochastic domains,". *Artificial Intelligence*, **101**:99-134 (1995).
18. Hassab Elgawi, O.: "Architecture of behavior-based Function Approximator for Adaptive Control,". In: *Proc. 15th Int'l. Conf on Neural Information Processing* ICONIP, LNCS 5507, pp.104-111 (2008).
19. Hassab Elgawi, O.: "Random-TD Function Approximator," In: *Journal of Advanced Computational Intelligence and Intelligent Informatics* (JACIII), 13(2):155-161 (2009).
20. Meuleau, N., Peshkin, L., Kim, K.-E., Kaelbling, L.: "Learning finite-state controllers for partially observable environments,". In: *Proc of the 15th Int'l Conf on Uncertainty in Artificial Intelligence*, pp.427-436 (1999).
21. Peshkin, L., Meuleau, N., Kaelbling, L.: "Learning policies with external memory,". In: *Proc. of the 16th Int'l Conf on Machine Learning*, pp.307-314, I. Bratko and S. Dzeroski, (Eds.) (1999)

22. Kenneth, O.: "Efficient evolution of neural networks through complexification,". Ph.D. Thesis; Department of Computer Sciences, The University of Texas at Austin. Technical Report AI-TR-04-314 (2004).
23. Gomez, F.: "Robust non-linear control through neuroevolution,". Ph.D. Thesis; Department of Computer Sciences, The University of Texas at Austin. Technical Report AI-TR-03-303 (2003).
24. Santamaria, J., Sutton, R., Ram, A.: "Experiments with reinforcement learning in problems with continuous state and action spaces,". In:*Adaptive Behavior*, **6**(2):163-218 (1998).

Appendix

8 Pole-balancing learning parameters

Below are the equations and parameters used for cart-pole balancing experiments [23] The equations of motion for N unjoined poles balanced on a single cart are

$$\ddot{x} = \frac{F - \mu_c sgn(\dot{x}) + \sum_{i=1}^{N} \tilde{F}_i}{M + \sum_{i=1}^{N} \tilde{m}_i}$$

$$\ddot{\theta}_i = -\frac{3}{4l_i}(\ddot{x}\cos\theta_i + g\sin\theta_i + \frac{\mu_{pi}\dot{\theta}_i}{m_i l_i}),$$

where $\tilde{F}_i$ is the effective force from the i^{th} pole on the cart,

$$\tilde{F}_i = m_i l_i \dot{\theta}_i^2 \sin\theta_i + \frac{3}{4} m_i \cos\theta_i (\frac{\mu_{pi}\dot{\theta}_i}{m_i l_i} + g\sin\theta_i),$$

and $\tilde{m}_i$ is the effective mass of the i^{th} pole,

$$\tilde{m}_i = m_i(1 - \frac{3}{4}\cos^2\theta_i).$$

Table 2 Parameters for the single pole problem.

Parameters for the single pole		
Symbol	Description	Value
x	Position of cart on track	$[-2.4, 2.4]$m
θ	Angle of pole from vertical	$[-12, 12]$deg
F	Force applied to cart	-10.10N
l	Half length of pole	0.5m
mc	Mass of cart	1.0kg
mp	Mass of pole	0.1kg

Table 3 Parameters for the double pole problem.

Parameters for the single pole		
Symbol	Description	Value
x	Position of cart on track	$[-2.4, 2.4]$m
θ	Angle of pole from vertical	$[-36, 36]$deg
F	Force applied to cart	-10.10N
l_i	Half length of i^{th} pole	$l_1 = 0.5$m
		$l_2 = 0.05$m
mc	Mass of cart	1.0kg
mp_i	Mass of i^{th} pole	$mp_1 = 0.1$kg
		$mp_2 = 0.01$kg
μ_c	friction coefficient on cart on track	0.0005
μ_p	friction coefficient if i^{th} pole's hinge	0.0005

9 Parameters for comparisons in cart pole balancing

Below are the parameters used to obtain the comparison result for SARSA-CABA, SARSA-CMAC, Q-MLP [23], and VAPS [20] in Section 6.3.

Table 4 Parameters for value function methods.

Parameter	Description
ε	greediness of policy
α	learning rate
γ	discount rate
λ	eligibility

Table 5 Parameters used for Q-LMP.

Parameter	Task		
	$1a$	$1b$	$2a$
ε	0.1	0.1	0.05
α	0.4	0.4	0.2
γ	0.9	0.9	0.9
λ	0	0	0

Table 6 Parameters used for SARSA-CABA.

Parameter	Task	
	$1a$	$1b$
Γd	0.03	0.03
Γ_k^x	0.05	0.05
Γ_k^x	0.1	0.1
ε	0.05	0.05
α	0.4	0.1
γ	0.99	0.99
λ	0.4	0.4

Table 7 Parameters used for SARSA-CMAC.

Parameter	Task	
	$1a$	$1b$
ε	0.05	0.05
α	0.4	0.1
γ	0.9	0.9
λ	0.5	0.3
No. of tilings	45 :	50 :
	10 based on $x, \dot{x}, \theta_1$	10 based on x_t, x_{t-1}, θ_t
	5 based on x, θ	10 based on $x, \theta_t, \theta_{t-1}$
	5 based on $x, \dot{\theta}$	5 based on x_t, θ_t
	5 based on $\dot{x}, \dot{\theta}$	5 based on x_{t-1}, θ_{t-1}
	5 based on x	5 based on x_t
	5 based on $\dot{x}$	5 based on x_{t-1}
	5 based on θ	5 based on θ_t
	5 based on $\dot{\theta}$	5 based on θ_{t-1}

A Fast Approximated Evolutionary Approach to Improve SVM Accuracy

Alessandro Perolini

Abstract Improving the classification performance is a crucial step of any machine learning method. In order to achieve a better classification Support Vector Machines need to tune parameters and to select relevant variables. To simultaneously perform both targets an embedded approach can be considered. This method consists of a two-layer algorithm where an evolutionary approach handles the solutions and an approximated one evaluates them. The evolutionary search, based on approximated error measures computed on the kernel matrix, allows discovering solutions which have high classification accuracy. The aim of the paper is to verify whether the proposed method is able to find reliable solutions which enhance the classification performance. The proposed method is applied on three real-world datasets using three kernels. In the experiments it is compared against the enclosed Genetic Algorithms and SVMs approach to demonstrate the ability of the approximated method to achieve high classification accuracy in a shorter time.

1 Introduction

This paper investigates a kernel matrix-based evolutionary search to assess the ability of this approximated technique to select the best subset of features and the optimal values of kernel's parameters. The importance of feature and model selection in a classification process is well known ([12], [22], [6], [7] and [14]). To ensure good predictions Support Vector Machines' (SVMs) classifiers have to be set up thus both procedures have to be performed. The relevance and the costs of executing a search of relevant features and good parameters' values of SVM's predictors and kernels induce researchers to suggest approaches like the Gradient-based method proposed by [1] and [7] and the span and features rescaling method advised by [6]

Alessandro Perolini
Politecnico di Milano, p.za Leonardo da Vinci, 32, 20133 Milano Italy,
e-mail: alessandro.perolini@polimi.it

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_14,

for the model selection problem, or the feature selection techniques summarized in [20] and [14]. The main drawback of these approaches is that they are performed separately, one by one. Evolutionary Algorithms (EAs) overcome the limitations of well-established methods providing a general framework which supports both procedures ([12], [16], [25] and [3]). In fact, they perform a simultaneous search of features and classifier's parameters allowing to reach high classification accuracy. Moreover, they reduce the risk of local optima falling preserving the solution from getting into a trap where a sub-optimal solution is placed. The EAs approaches are usually developed through a traditional search ([16], [25] and [3]) which requires to train a SVM or through faster methods like [12] which uses bounds.

Although a joined EA-SVM approach improves the performance of SVMs it is expensive. In fact, for all the chromosomes of a population a complete training must be performed. This means that for each generation as many optimization problems as the number of individuals have to be solved. To deal with this drawback, instead of using a time consuming method, a kernel matrix approach can be considered. The embedded kernel matrix criteria and evolutionary algorithm method uses kernel matrix indicators to lead the evolution to the best solution.

The paper is organized as follows. Sect. 2 describes feature and model selection problems highlighting the reasons of a simultaneous search. Sect. 3 introduces the EA-SVM approach and analyzes the kernel matrix method. Sect. 4 compares the research methods summarizing the experiments' results on training and test sets.

2 Feature and Model Selection

Referring to a SVM classification task feature and model selection are applied in order to identify relevant features and to find the best values of kernel's and SVM's parameters. Both things influence the classification process either improving or reducing the classifier's predictive ability ([12], [22], [14] and [19]).

2.1 Problem Overview

Feature selection is applied with the purpose of choosing relevant and rejecting irrelevant and redundant variables while model selection looks for the parameters' values of SVM and kernel to improve classifier's performance. Many researches investigate feature selection methods showing the drawbacks of using non-relevant attributes and focusing the attention on the effects of those methods on selection process [20], [14] and [12]. Conversely, authors like [11] and [22] examine the model selection problem pointing out the impacts on classification methods performance.

Kohavi [20] summarizes the feature selection process in filter and wrapper approaches. Guyon and Elisseeff [14] discuss several selection procedures highlighting their strengths and weaknesses. Among their considerations they present two

important remarks about filter and wrapper techniques. To assess attributes filters use independent measures that do not involve the learning machine. On the contrary, since wrappers require a classification method (e.g. SVM) to select features, the goodness of the features' subset depends on the classifier's performance. As a consequence, attributes are "good" if the classifier performs well. But the performance of a classifier depends, in turn, on the employed parameters. Thus, for each subset of features a model selection procedure is required. This means that feature and model selection are strictly connected each other. Other researchers [11], [22] and [12] provide similar observations. Duan et al. [11] introduce the problem of parameters tuning comparing several performance measure like span bound, VC bound, radius margin bound. Rakotomamonjy [22], studying SVM bounds for ranking features, asserts that hyperparameters is "a crucial issue". Hence, to achieve the best generalization performance, classifier's parameters have to be tuned. Fröhlich et al. [12] state "[...] we are trying to select an optimal feature subset and an optimal C at the same time. This is reasonable, because the choice of the parameter C is influenced by the feature subset taken into account and vice-versa". This consideration can be easily extended to kernel's parameters. [20], [14], [12], [22] and [11] underline, through different points of view, the importance of tuning model's parameters and to select relevant variables in order to reach high classification accuracy. In particular, [12], [22] and [14] raise the issue of a joined research.

3 GA-based Method

To deal with the problem of a combined search of parameters' values and features a Genetic Algorithms (GAs) approach is considered. GAs are able to perform feature and model selection simultaneously providing more benefits rather than applying these techniques in sequence. The GA and SVM joined process answers the remarks described in Sect. 2.1. In fact, SVM can be embedded in the GA process using its results in the fitness function computation. The performance reached by the classifier defines the fitness value of each chromosome that composes the population. GA-SVM method shows considerable results on real-world datasets [12] and it has been successfully applied in several contexts, from informatics [3] to biomedical field [16] or to financial one [25].

3.1 Genetic Algorithms and Support Vector Machines

Without loss of generality, a binary classification problem can be considered. It consists of a dataset of m points $(x_i, y_i), i \in M = \{1, \ldots, M\}$ in $\mathbb{R}^{n+1}$ where x_i is a n-dimensional vector and y_i is a scalar that represents the class of the *i-th* instance where $y_i \in \{-1, +1\}$.

3.1.1 Genetic Algorithms.

GA is a heuristic search technique for optimization problem originally developed by [15] and [13]. It is based on the Darwinian principle of natural selection: individuals adapted to the environment survive the natural selection while the others don't. GAs reproduce the natural selection making a set of q individuals (i.e. a population) evolve in order to improve their survival skills. The GA-SVM formulation requires that each individual is defined as a $n + p$-dimensional vector ($chrom \in H$ where H is the space of individuals) where n is the number of variables and p is the number of kernel's parameters[1]. The variables's part is represented by a binary codification while the parameters' part has a real representation, see Fig. 1.

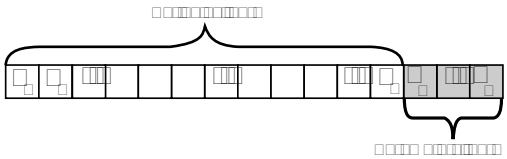


Fig. 1 Chromosome's structure

The goodness of an individual is summarized by the fitness value (f_v) that is, in general, the accuracy – or the error – of SVMs' classifier. Individuals that show good skills (fitted chromosomes) survive the artificial selection and will contribute to create the population of the next generation. Every generation a set of μ parents are selected and put in the matting pool for reproduction. The reproduction phase generates μ offspring that are included in the population replacing the parents. Unfitted chromosomes are mutated and then included in the population while the best chromosomes are passed to the next generation using the elitism strategy. The evolution continues until the population converges to a solution or some stopping criteria end the search.

Three operators control the evolution: selection, crossover and mutation operators. The selection operator, $selec : H^q \rightarrow H^\mu$, selects μ individuals – from q individuals of the population – for the reproduction. The crossover operator, $cross : H^\mu \rightarrow H^\mu$, performs the genes exchange considering the q selected individuals while the mutation operator, $mut : H^{q-\mu} \rightarrow H^{q-\mu}$, introduces small variations to genes' values for the $q - \mu$ remaining individuals. Moreover, in order to preserve the optimal solutions, the elitism strategy is applied passing to the next generation p_e chromosomes which show high fitness value. In the "standard formulation" of GA-SVM method, the search for the optimum is performed maximizing the classifier's accuracy or minimizing its error. Thus, the computational time of a complete GA process depends, besides data, kernel type and algorithm chosen to solve SVM optimization problem, on the training procedure (k-folds Cross Validation or LOOCV), on the number of individuals that compose the population and on the stopping rules.

[1] The number of kernel's parameters depends on the considered kernel.

3.2 A Kernel Matrix-based Approach

Since training a SVM is an expensive procedure, several alternatives, like those proposed by [23], [12], [22] and [7] are considered to deal with the optimization process. Besides these options, kernel matrix criteria are suitable for optimizing kernel's parameters and selecting relevant attributes. They assess the kernel matrix – also known as Gram matrix – with the purpose of estimating the classifier's error. In order to describe the role of the kernel matrix in SVM-based classifiers a brief introduction on the SVM optimization problem is provided in the following section.

3.2.1 SVMs Optimization Problem.

SVM is a supervised learning classification method developed by [23] and [24] in the Statistical Learning Theory context and linked with the previous literature through the optimal margin classifier by [2]. SVM classifiers discriminate instances recognizing the patterns behind data. The underlying hypothesis is that similar patterns belong to the same classes.

Referring to the dataset described in Sect. 3.1 the classification process identifies observations that belong to positive and negative classes. This, for the linear case, is obtained computing the hyperplane ($w \cdot x + b = 0$, $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$) which provides the largest separation – the optimal margin hyperplane – between instances. Observations are then separated computing the sign of the decision function $f(x) = sign(w \cdot x + b)$, using the values of w and b provided by the solution of the optimization problem (1), and assigning to the i-th istance a class label.

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ & y_i(w \cdot x_i + b) \geq 1 \qquad i = 1, \ldots, m \end{aligned} \tag{1}$$

Unfortunately this problem is referred to a classifier which provides zero error. Thus, in order to allow misclassified instances the constraints of (1) have to be relaxed introducing slack variables ξ_i. The optimization problem becomes as (2) that corresponds to the Lagrangian formulation (3).

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i \\ & y_i(w \cdot x_i + b) \geq 1 - \xi_i \qquad i = 1, \ldots, m \\ & \xi_i \geq 0 \qquad i = 1, \ldots, m \end{aligned} \tag{2}$$

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j x_i x_j \\ & \sum_{i=1}^{m} y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C \qquad i = 1, \ldots, m \end{aligned} \tag{3}$$

Where α_i are the Lagrange multipliers.

Anyway, if data are not linearly separable the Lagrangian formulation (3) of the dual quadratic optimization problem can be extended to the nonlinear case through kernel functions. This can be done because data appears as a dot product between vectors [2].

Before introducing the nonlinear SVM two definitions are required. A kernel function k is a function such that $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$ for all $x_i, x_j \in X$ where $\phi(x) : X \to F$. It[2] maps data from an input space (X) to a higher dimension feature space (F) where a linear separation can be performed [10]. A kernel matrix (4) is a positive semi-definite matrix that respects the Mercer's condition. It is defined as

$$K_{ij} = \{\langle \phi(x_i), \phi(x_j) \rangle\}_{i=1,\ldots,m; j=1,\ldots,m} \tag{4}$$

Thus switching to the nonlinear case is done substituting the dot product ($\langle x_i, x_j \rangle$) of the linear formulation (3) with the kernel matrix (4) obtaining the objective function

$$\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j K(x_i, x_j)$$

A detailed description on SVM optimization problems and kernel functions are reported in [4] and [10].

3.2.2 Performance's Estimation Through Kernel Matrix.

The nonlinear optimization problem, described in the previous paragraph, highlights two key factors that influence the SVM classifier's performance: the SVM's parameter (C) and the kernel matrix ($K(x_i, x_j)$). In order to improve the classification performance both elements have to be chosen carefully. This means finding the values that provide the best prediction on the training set. Usually, both factors are investigated through a grid search or solving an optimization problem ([6] and [7]) or using an evolutionary search ([3] [16] [25]) that requires to train a classifier. Anyway, all these processes entail a wasting of time that can be reduced analyzing the kernel matrix. Since it holds all the information required by the classifier, the kernel matrix becomes a good indicator of the classifier's performance. Therefore the estimation of the prediction's error can be computed just considering the kernel matrix.

[2] Kernel functions considered in the paper are reported in Sect. 4

Even if this procedure increases the speed of the GA evolution the main drawback of the kernel matrix-based method is that it does not allow handling the parameter that controls the generalization ability and the complexity of the SVM (C), thus only kernel parameters' values will be searched. In order to verify the true performance of the benchmark the C parameter will be fixed to an appropriate value nor too small to prevent underfitting neither too big to avoid a complex model and so overfitting problem ([8], [10] and [18]).

3.2.3 Kernel Matrix Criteria.

As described in the previous paragraph the classifier's performance can be assessed by means of the kernel matrix. Therefore the classification's error can be estimated through a "kernel matrix goodness indicator". This measure replaces the error attained through the SVM training providing to the GA cycle the fitness value of population's chromosomes. Three kernel matrix criteria are considered: the Kernel Target Alignment (KTA) suggested by [9], the Feature Space-based kernel matrix evaluation Measure (FSM) introduced by [21] and the Feature distance based Combinatorial kernel Matrix evaluation Criterion (FCMC) developed by [17]. The purpose of the mentioned kernel matrix criteria is, for the KTA, to estimate the classifier's error discovering the agreement between the kernel matrix and the target matrix and, for FSM and FCMC, capturing patterns' differences hold in the kernel matrix with respect to positive and negative classes.

The KTA [9] is based on a simple principle: assessing the classification's performance using the cosine between two kernel matrices as similarity measure. In the SVM context this indicator can be employed in order to estimate the classifier's accuracy computing the alignment between the kernel matrix and the target matrix T. The target matrix T is defined as $T = y_i \cdot y_i'$ where y_i is the vector of class' labels. Crisitanini et al. [9] prove that it is possible to bound the error using the empirical alignment and define the KTA as the normalized Frobenius inner product between kernel and target matrices

$$KTA = \frac{\langle K, T \rangle_F}{\sqrt{\langle K, K \rangle_F \langle T, T \rangle_F}} \tag{5}$$

High values of KTA indicates a good kernel choice thus good expected classification performance.

Before introducing FSM and FCMC criteria a brief remark is required. It is convenient to sort the dataset positioning positive class instances at first rows and the negative ones afterwards. Thus the number of instances that belong to positive and negative classes are m_+ and m_-, respectively.

The FSM [21] appraises the classifier's error relaxing some hypoteses of KTA criterion. It can be seen as an indicator that measure the separation between classes, in fact, it is defined (6) as the ratio of the total within class variance in the direction between the class centers to the distance between the class centers.

$$FSM = \frac{std_+ + std_-}{\|\phi_- - \phi_+\|} \tag{6}$$

Where the within class standard deviations in the direction between class centers are $std_+ = \sqrt{\frac{\sum_{i=1}^{m_+} \langle \phi(x_i) - \phi_+, e \rangle^2}{m_+ - 1}}$ and $std_- = \sqrt{\frac{\sum_{i=m_++1}^{m} \langle \phi(x_i) - \phi_-, e \rangle^2}{m_- - 1}}$, the class centers are $\phi_+ = \sum_{i=1}^{m_+} \phi(x_i)/m_+$ and $\phi_- = \sum_{i=m_++1}^{m} \phi(x_i)/m_-$ and $e = (\phi_- \phi_+)/\|\phi_- \phi_+\|$.

The FCMC [17] measures the classifier's error assessing the similarity of features that belong to the same class respect to the distance between classes. It is defined as

$$FCMC = \frac{d_{in}^+ + d_{in}^-}{d_{out}} \tag{7}$$

where the sum of the distances between features and their centers within a class are $d_{in}^+ = \sum_{i=1}^{m_+} \langle \phi(x_i) - \phi^+, \phi^+ + \phi^- \rangle^2$ and $d_{in}^- = \sum_{i=m_++1}^{m} \langle \phi(x_i) - \phi^-, \phi^+ + \phi^- \rangle^2$ and the distance between classes is $d_{out} = \|\phi^+ + \phi^-\|^2$.

3.2.4 Error's Measures for Kernel Matrix Criteria.

Each kernel matrix criterion provides a value that is not directly comparable with the other indicators. In fact, KTA estimates the classifier's accuracy in a $[0,1]$ range while FSM and FCMC estimate the quality of the kernel matrix using a $[0,\infty)$ range. Therefore, in order to compare the criteria, for each indicator an error measure is provided:

$$KTA_{err} = 1 - KTA$$
$$FSM_{err} = \frac{FSM^2}{1 + FSM^2}$$
$$FCMC_{err} = \frac{FCMC^2}{1 + FCMC^2}$$

KTA_{err}, FSM_{err} and $FCMC_{err}$ indicators vary in a $[0,1]$ range making the comparison easier. A low indicators' value suggests a good kernel matrix hence a low expected classifier's error.

4 Experiments

In order to evaluate the kernel matrix usage in a GA search some experiments were conducted on three datasets taken from the UCI Machine Learning Repository: Breast cancer Wisconsin, Credit approval and German Statlog. Datasets's details are reported in Table 1. Kernel matrix measures were compared in terms of error on test set with the purpose of evaluating the trade off between achieved performance and loss of information caused by the use of a rough technique.

Experiments are conducted using the GA-SVM method (as baseline) and the kernel matrix-based one. Both approaches are based on the scheme described in Sect. 3 but they use a different fitness value: the error of the classifiers for the baseline method and the estimated error for the proposed one. The solution of the search consists of two parts: a subset of features and the values of kernel's parameters. It corresponds to the classifier that achieves the lowest classification error on the training set. In order to get more reliable results experiments are performed ten times each. Each experiment consists of two steps:

1. ***Best classifier's identification.*** A method (proposed or baseline[3]) is applied on the training set to find the solution with the lowest error (this is considered as the best);
2. ***Classifier's evaluation on test set.*** The classifier that corresponds to the best solution is applied on the test set and the error is computed.

Before performing the trials a preprocessing phase was applied to all datasets: numerical variables were normalized in a $[0,1]$ range while categorical ones were transformed in dummies. In addition, for Breast cancer and Credit approval datasets, missing values were removed. After the preprocessing phase, datasets were split in training and test sets with a percentage of instances of 70% and 30%, respectively. Experiments were conducted using a not optimized Matlab implementation of Genetic Algorithms (with Matlab code functions for kernel matrix criteria and libsvm [5] for GA-SVM approach) considering three kernel types: linear ($K(x_i,x_j) = x_i \cdot x_j$), RBF ($K(x_i,x_j) = exp^{-\|x_i - x_j\|^2/2\sigma^2}$) and sigmoid ($K(x_i,x_j) = \tanh(\sigma(x_i \cdot x_j) + \theta)$).

Table 1 Datasets description.

Dataset	Tot. inst.	Instances		Var.	Numerical	Categorical
		Train	Test			
Breast	683	478	205	9	9	–
Credit	690	483	107	15	6	9
German	1000	700	300	20	7	13

All trials had the same GA settings. The size of the population was set to 80 and the maximum number of generations, considered as stopping rule, was set to 80 too. The starting population (i.e. the set of solutions of the GA) was randomly initialized. The tournament selection operator had a number of players equal to 60, the n-points crossover and the uniform mutation operators were applied with a probability of 75% and 5%, respectively. The elitist strategy was used to preserve the best solution using 15% of the population's chromosomes. The values of kernel's parameters vary in predefined ranges: for the RBF kernel the range of σ was $(0,10]$ while for the Sigmoid kernel the range of σ was $(0,10]$ and of θ was $[0,10]$.

[3] For the baseline method a 10-folds cross validation is used.

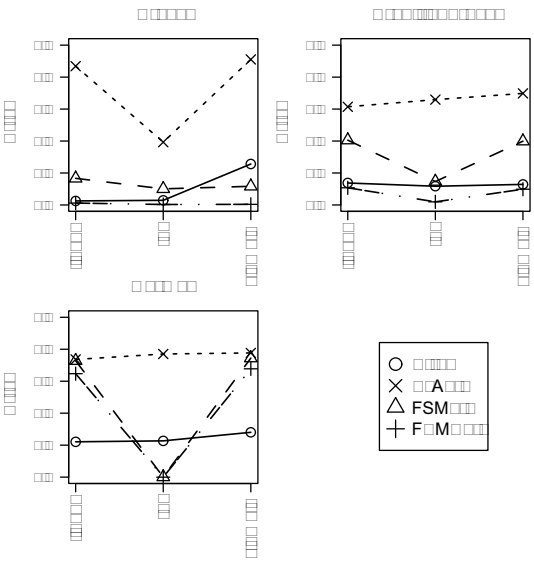


Fig. 2 Mean classifiers' performances on training set – Cross Validation error and kernel matrix estimated errors

4.1 Experiments – Behavior on Training Set

The results in term of estimated error of KTA_{err}, FSM_{err} and $FCMC_{err}$ (Fig. 2) show how kernel matrix criteria are able to reach, on the training set, comparable classification performance of GA-SVM error. Sometimes they outperform the GA-SVM method, like FCMC for Breast cancer and Credit approval datasets, while, other times, they fail recognizing the lowest values, has it happens for the German dataset. Moreover, kernel matrix criteria present different behaviors: FSM and FCMC show closest estimated errors, but disclose more variability; on the contrary, KTA presents steady estimated errors far from the best GA-SVM.

4.2 Experiments – Classifiers' Performances on Test Set

Table 2 presents the best solutions of the evolutionary process based on kernel matrix criteria comparing them to the best solutions found by GA-SVM approach. Among all runs, the solution that has the lowest error on test set is considered as best, but if more than one reach the lowest value the number of selected attributes[4] is used as second choice criterion.

Comparing the results in terms of kernels and datasets it is possible to state that on Breast cancer dataset only FCMC is able to discover a good subset of variables that allows to perform better than the best solution of GA-SVM. Moreover, even if KTA is not able to reach good results with RBF kernel, on the sigmoid one it

[4] Similar results are reported in Table 3.

Table 2 Experiments results - best solutions on test set. Bold values indicate kernel matrix criteria's errors on test set that outperform the GA-SVM method, while bold-italic values indicate similar results. Moreover, boldface selected variables point out the lowest number of features. Both comparisons are done with regard to kernels.

	Crit.	Kernel	Error		Sel.	Parameter		
			Train	Test	var.	C	Sigma	Theta
Breast	Error	linear	0.0272	0.0293	**5**	1	–	–
	FCMC	linear	0.0125	**0.0244**	6	1	–	–
	Error	rbf	0.0251	0.0293	6	1	0.030772	–
	KTA	rbf	0.3931	0.2049	6	1	4.2414	–
	Error	sigmoid	0.0272	0.0195	**8**	1	0.0025	1.234
	KTA	sigmoid	0.9137	***0.0341***	9	1	0.00038	0.71744
Credit	Error	linear	0.1379	0.1276	21	1	–	–
	KTA	linear	0.6148	**0.1276**	**8**	1	–	–
	Error	rbf	0.1116	0.1122	22	1	1.522	–
	KTA	rbf	0.6773	***0.1174***	**14**	1	0.72464	–
	Error	sigmoid	0.1357	0.1276	20	1	0.19008	0.1515
	KTA	sigmoid	0.7207	**0.1225**	**16**	1	0.44861	0.000549
German	Error	linear	0.2243	0.2667	**34**	1	–	–
	FCMC	linear	0.6480	***0.2733***	48	1	–	–
	Error	rbf	0.2400	0.2567	38	1	0.12898	–
	KTA	rbf	0.7758	***0.2667***	**25**	1	1.1636	–
	Error	sigmoid	0.2757	0.2833	34	1	5.3335	1.5249
	KTA	sigmoid	0.7516	***0.2867***	**17**	1	0.5172	0.052184

performs as well as GA-SVM error. On Credit approval dataset KTA performs well with all kernels: with linear and RBF ones it reaches similar results of the benchmark, while, it overcomes the best GA-SVM model with the sigmoid one. On German dataset FCMC and KTA perform as well as GA-SVM: in fact, the errors on test set are close to each other.

Some additional remarks can be made. Comparing Table 2 and 3, it can be shown that FSM and FCMC select roughly the same number of attributes, while KTA is able to reduce it very much. Beyond the best solutions summarized in Table 2, it is interesting to analyze the "second best" results of Table 3, where the same classification's errors are obtained by different criteria. Achieved results prove that the evolutionary process based on approximated measures reaches low error's values providing multiple optimal solutions for the same dataset – see the Credit approval dataset with linear kernel. These experiments confirm the remarks of [12], [22] and [14], which state that there is a relation between variables and employed parameters.

The analysis of computing time (Fig. 4) reveals that kernel matrix-based criteria employ a small time if compared with GA-SVM and get better performances. In particular, on more complex classification problems (like Credit and German datasets) they provide the highest reduction of time. On the Breast cancer dataset the time performances of kernel matrix criteria are quite the same of GA-SVM, excepts for

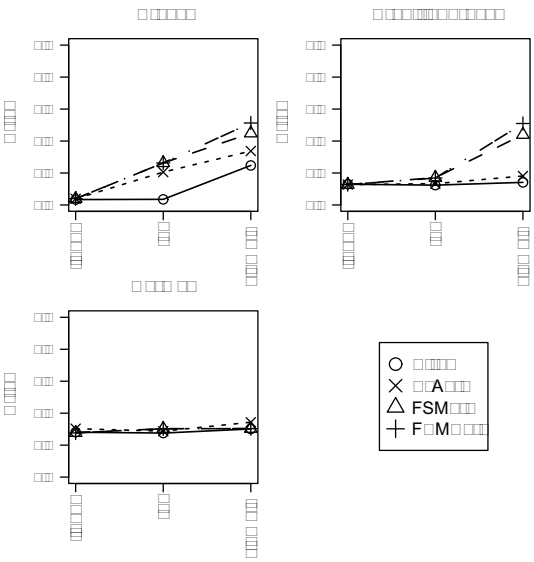


Fig. 3 Mean classifiers' performances on test set

the sigmoid kernel, which requires higher efforts. This means that the use of an approximated approach to deal with feature and model selection problem was really useful. However, the results on the last two datasets point out a different conclusion. On Credit dataset the reduction of time is clearly high, kernel matrix-based criteria evolutions provide, with a lower effort, better results than GA-SVM one (see Table 2). Moreover, German dataset results confirm the reduction of time and the low classification error obtained by the proposed method.

Table 3 Similar classification performances of kernel matrix criteria on test set. Bold values indicate kernel matrix criteria's errors on test set that perform better than GA-SVM method while, bold-italic values indicate comparable results.

	Crit.	Kernel	Error		Sel.	Parameter		
			Train	Test	var.	C	Sigma	Theta
Credit	FCMC	linear	0.1078	**0.1276**	29	1	–	–
	FSM	linear	0.4041	**0.1276**	26	1	–	–
	KTA	linear	0.6148	**0.1276**	8	1	–	–
	FCMC	rbf	0.0004	***0.1327***	22	1	0.004888	–
	FSM	rbf	0.0218	***0.1327***	24	1	0.004296	–
German	FCMC	linear	0.6480	***0.2733***	48	1	–	–
	FSM	linear	0.7287	***0.2733***	48	1	–	–
	FCMC	sigmoid	0.6650	*0.3033*	32	1	0.0042651	8.558
	FSM	sigmoid	0.7566	*0.3033*	29	1	0.0034479	0.43149

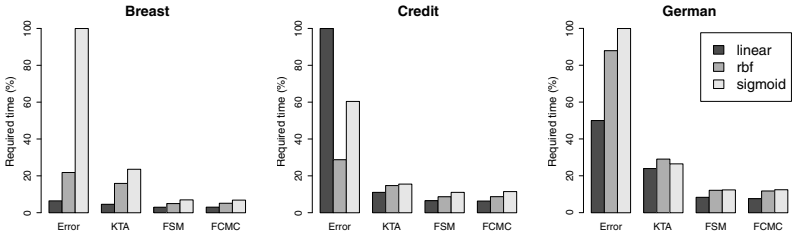


Fig. 4 Mean required time for a complete evolution

5 Conclusion

In this paper the joined kernel matrix criteria and Genetic Algorithm approach was proposed to deal with simultaneous feature and model selection problem for SVM-based classifiers. Through the experiments it was proved that kernel matrix-based evolutionary approach provides an effective search method to select the best subset of features and the optimal values of kernel's parameters.

The proposed method shows remarkable strengths. It does not require neither to set the number of features to retain nor to introduce restrictive hypotheses, unlike other optimization methods do. It reduces the problem of local optima falling using a GA search process. Additionally, it represents a valid alternative to GA-SVM method in terms of computational time required and optimal solutions reached. On the one hand, if compared with GA-SVM approach, the time employed to complete a whole evolution is shorter making it computationally attractive. On the other hand, with regards of classification performances, it shows high effectiveness. In fact, on the test set the classification's errors obtained by the approximated method were better or, at worse, close to the best solutions of GA-SVM approach. Kernel matrix criteria were able to outperform GA-SVM three times out of six and performed the same once (see Table 2). Among kernel matrix criteria, classifiers trained with the settings provided by KTA showed the best predictions: on test set KTA was able to perform better than FSM and FCMC. However, the last two criteria confirm their effectiveness providing comparable errors' values.

References

1. Bengio, Y.: Gradient-based optimization of hyperparameters. Neural computation **12**(8), 1889–1900 (2000)
2. Boser, B.E., Guyon, I.M., Vapnik, V.N.: Training algorithm for optimal margin classifiers. In: Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory (1992)
3. Braga, P.L., Oliveira., A.L.I., Meira, S.R.L.: A ga-based feature selection and parameters optimization for support vector regression applied to software effort estimation. In: Proceedings

of the 23rd Annual ACM Symposium on Applied Computing, SAC'08, pp. 1788–1792. Association for Computing Machinery (2008)
4. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery **2**(2), 121–167 (1998)
5. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001)
6. Chapelle, O., Vapnik, V.: Model Selection for Support Vector Machines (2000)
7. Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S.: Choosing multiple parameters for support vector machines. Machine Learning **46**(1-3), 131–159 (2002)
8. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3), 273–297 (1995)
9. Cristianini, N., Kandola, J., Elisseeff, A., Shawe-Taylor, J.: On kernel-target alignment. In: Advances in Neural Information Processing Systems 14, vol. 14, pp. 367–373 (2002)
10. Cristianini, N., Shawe-Taylor, J.: An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press (2000)
11. Duan, K., Keerthi, S.S., Poo, A.N.: Evaluation of simple performance measures for tuning svm hyperparameters. Neurocomputing **51**, 41–59 (2003)
12. Fröhlich, H., Chapelle, O., Schölkopf, B.: Feature selection for support vector machines by means of genetic algorithms. In: Proceedings of the 15th IEEE International Conference on Tools with artificial Intelligence, pp. 142–148 (2003)
13. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc, Boston, MA, USA (1989)
14. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research **3**, 1157–1182 (2003)
15. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI, USA (1975)
16. Huang, H.L., Chang, F.L.: Esvm: Evolutionary support vector machine for automatic feature selection and classification of microarray data. BioSystems **90**(2), 516–528 (2007)
17. Jia, L., Liao, S.: Combinatorial kernel matrix model selection using feature distances. In: Proceedings of International Conference on Intelligent Computation Technology and Automation, ICICTA 2008, vol. 1, pp. 40–43 (2008)
18. Joachims, T.: Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms. Kluwer Academic Publishers, Norwell, MA, USA (2002)
19. Kira, K., Rendell, L.A.: Feature selection problem: traditional methods and a new algorithm. In: Proceedings 10th National Conference on Artificial Intelligence - AAAI-92, pp. 129–134 (1992)
20. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence **97**(1-2), 273–324 (1997)
21. Nguyen, C.H., Ho, T.B.: An efficient kernel matrix evaluation measure. Pattern Recognition **41**(11), 3366–3372 (2008)
22. Rakotomamonjy, A.: Variable selection using svm-based criteria. Journal of Machine Learning Research **3**, 1357–1370 (2003)
23. Vapnik, V.: The nature of statistical learning theory. Springer-Verlag New York, Inc (1995)
24. Vapnik, V.N.: Statistical Learning Theory. Wiley, New York (1998)
25. Wu, C.H., Tzeng, G.H., Goo, Y.J., Fang, W.C.: A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy. Expert Systems with Applications **32**(2), 397–408 (2007)

EVOLUTIONARY ALGORITHMS, BAYESIAN NETWORKS AND MODEL-BASED DIAGNOSIS

A Particle Swarm Optimization Approach for the Case Retrieval Stage in CBR

Nabila Nouaouria and Mounir Boukadoum[1]

Abstract Finding the good experiment to reuse from the case memory is the key of success in Case Based Reasoning (CBR). The paper presents a novel associative memory model to perform this task. The algorithm is founded on a Particle Swarm Optimization (PSO) approach to compute the neighborhood of a new problem. Then, direct access to the cases in the neighborhood is performed. The model was experimented on the Adult dataset, acquired from the University of California at Irvine Machine Learning Repository and compared to flat memory model for performance. The obtained results are very promising.

1 Introduction

CBR's problem solving methodology is founded on reusing the solutions of past problems to solve new similar ones. Past situations and their solutions are stored in a case memory, and finding a good experiment to reuse, one that can lead to adequate inferences once retrieved, is the key of success for the reasoning. This is the recall process and it is highly influenced by memory organization and by retrieval strategies. As a result, the accuracy (in the sense of exhaustiveness) and speed of the recall task are two important parameters for the performance evaluation of a CBR system.

CBR can also be synergistically combined with other artificial intelligence tools [1]. Among the possible combinations, we present in this paper an approach to perform a quick recall in an associative memory, using swarm intelligence. The main idea is to compute the neighbourhood of a new problem with the PSO algorithm and, then, directly attain the contents of the neighbourhood via an associative memory mechanism. No prior organization of the case base is required for the process to take place.

1 Department of Computer Science, University of Quebec at Montreal
CP. 8888, Succ. Centre-ville, Montréal, QC H3C3P8 Canada
nouaouria.nabila@gmail.com, boukadoum.mounir@uqam.ca
phone. (514) 987-3000 p./ext. 4565# URL: www.info2.uqam.ca/~boukadou

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_15,

In the balance of this article, we start the next section by skimming over the retrieval stage in the CBR cycle (Section 2.1) and, then, describe the proposed memory model (Section 2.2). The used PSO approach is presented in Section 3, and Section 4 presents our results and discussion thereof. Finally, Section 5 concludes the work and presents perspectives for future research.

2 Case Retrieval Stage

"Reasoning is remembering" is the slogan of most researchers involved in CBR. In order to function correctly, CBR uses cases stored in a case base that is representative of the problems encountered in the field. The more cases the case base contains, the more likely it is to find a case for the reasoning that is similar to the new case. Thus, the quality of obtained solutions generally improves with the number of stored cases. However, this is achieved at the expense of computation cost which increases concomitantly. This makes efficient techniques of memory organization and search particularly important for the success of CBR.

2.1 Theoretical Background

There exist several memory organizations, for each one a search algorithm is used. Those can be divided into:

The flat memory/brute force algorithm: cases are stored sequentially in a simple list, array or file [2]. Cases will be retrieved by sequentially applying a matching function to each case in the file and keeping track of the matching score; the case with the best match is returned. There is no particular organization of the cases and the retrieval algorithm is quite simple since the matching heuristics do all the work. The major advantage of this memory organization is that the entire case library is searched. As a result, the accuracy of retrieval is only a function of how good the match function is. Moreover the addition of a new case is not expensive; but the memory organization becomes costly when the base is too large. To remedy this disadvantage, we can use alternatives such as surface indexes to reduce the total number of candidates, or partitions; or also, parallel implementations.

The structured memory/index based algorithm: Here, CBR memories are rich with a variety of generalized structures such as concepts, prototypes, and abstract cases. The accumulation of generalizations or abstractions facilitates the evaluation of the situation, and allows control of indexation. These structures can be organized via conceptual hierarchies, decision trees, object oriented taxonomies, formal concept lattices, and B-trees [3]. We also retrieve in this category Shared-Feature Networks where cases presenting similarities in the same

cluster are gathered and hierarchies are formed as the clusters are subdivided into smaller clusters and, dually, Discrimination Networks where a discrimination of cases is made as a side effect of clustering in shared-features networks [4].

All memory models that use a top-down search share two desirable features [5]: Data structuring by regrouping related objects and efficient retrieval by traditional tree search algorithms. Unfortunately they also have potential limitations, of which memory traversal by answering an ordered sequence of internal node questions (in the case of incomplete information, this could lead to erroneous paths) and difficult access to neighboring clusters with similar cases when reaching a cluster at a tree leaf. Two notable exceptions in the category of index-based approaches are the Fish & Shrink model [6] and the CRN model [5].

Schaaf, in [6], introduces a data structure to hold case representations (aspects) and links to store aspect specific similarities between cases. The Fish & shrink model is based on the concept of a polyhedral that represents a case. A face of it then corresponds to an aspect representation and the label of an edge to a calculated distance between two cases with respect to the connected aspect representations. A case base can be seen as a network of cases. Weighted edges from face to face connect cases. The weight depends on the distance of the connected cases with respect to a certain aspect. Two cases are called neighbors with respect to an aspect if they are connected by an edge concerning this aspect. Edges can be directed if the distance is not symmetric. The author explains how changes in point of view on cases can be seen as a spontaneous and weighted combination of aspects. This leads to the possibility of a context-dependent redefinition of case similarity by using only low cost calculations.

Lenz & al.'s CRN (for Case Retrieval Net) memory model [5] uses spreading activation and information completion. The CRN formalism offers significant retrieval speedup in comparison to linear search and has been successfully deployed over large case bases with as many as 200 000 records. The foundation of CRN is inspired from neural network techniques and associative memory models and it has remarkable features [7], of which a case recall process that does not browse a path in a tree. In CRN, retrieval is made in a reconstructive way by recovering information entities and gradually rebuilding the case. The most fundamental items in the context of CRN are information entities (IE), which represent knowledge items with the lowest level of granularity [7]. A case is a set of IEs and a case memory is a net of nodes corresponding to the IEs observed in the domain and additional nodes denoting the particular case. The IE nodes are connected by similarity arcs weighted by a similarity function, and a case node is reachable from its constituting IE nodes via relevance arcs enabled by a binary function. An improvement of this model is presented in [18]. The model considers both the similarity and adaptability criteria, with a subsequent improvement in recall accuracy for nearly the same computational effort. The targeted extension lies not in CRN's fundamental mechanisms but their utilization. In CRN, we only transport and collect similarity knowledge. In the proposed extension, we transport and collect both similarity and adaptability.

Table 1 summarizes the above retrieval methods and indicates their computational complexities. Notice that the computational complexities are about the retrieval stage and do not include the construction cost of the memory structure. This cost could also be very important, especially for the two last techniques.

Table 1. Overview of retrieval methods.

Type	Methods	Used for	Computational Complexity
Brute force	Sequential Search	Small case bases, simple Similarity	Depends on case base size n : O(n)
	Kd-Tree	Few attributes, Large case bases	Depends on tree depth
Index Based	Fish & Shrink	Complex similarity, Small Case bases	Depends on aspects connectivity and query size
	CRN	Few numerical attributes, Large case bases	Depends on query size, IE connectivity degree, and IE specificity.

In what follows, we propose a memory model that has the particularity of not requiring a prior case organization in memory. This leads to two major results: 1) An improvement of retrieval accuracy since no organization related bias is introduced during a pre-filtering phase; 2) a reduction of computational effort during the construction of case memory as will be seen.

2.2 Proposed Architecture

We propose a new vision of the retrieval problem that is based on constructing a problem neighborhood. The retrieval of applicable cases can be formulated as the extraction from the search space of a sub-space of cases that are similar to the problem to resolve. We call this sub-space the neighborhood of the target problem. In our approach, it is obtained by a PSO search strategy. We start from the flat memory structure of the case memory and construct a nested structure with two types of nodes: value nodes and case nodes. A value node, or Information Entity node, represents a particular value for a problem attribute (see figure 1). It is linked to all case nodes where it occurs. The case node points out to the case base location where the whole case is stored. In Figure 1 for example IE_2 and IE_a pointing to case node #1 mean that Case #1, which is pointed by case node #1, has IE_2 as first attribute value and IE_a as second attribute value.

The particularity of the proposed structure is that we reach a case by its contents (the principle of associative memories). Another particularity is that the structure could easily and automatically be built by simply parsing the case memory and constructing lists of value nodes and case nodes during a pre-treatment stage.

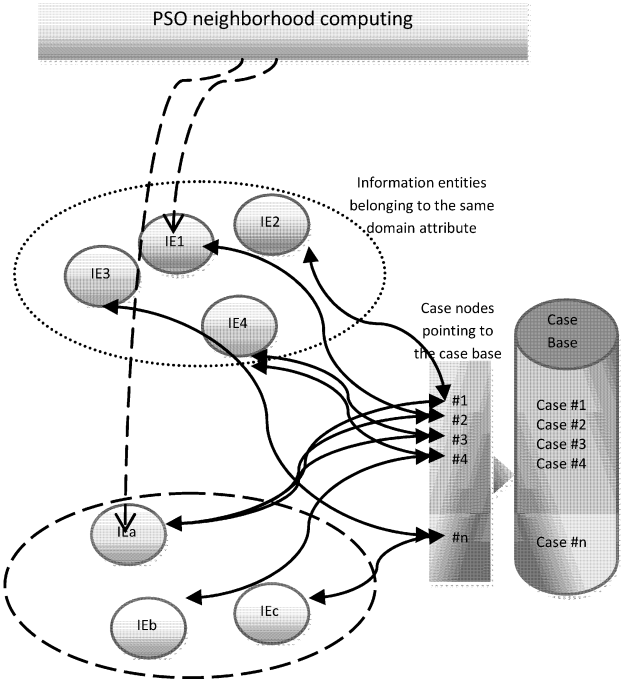


Figure 1 The proposed memory architecture.

Thus, each attribute domain is represented by a set of value nodes. Each case node is pointed by a set of value nodes, each of them belonging to a specific attribute domain. Each case node points out to the corresponding case in the case base. For instance, a query specifying the sub-set of value nodes= $\{IE_4, IE_a\}$ will have as result cases pointed by case nodes belonging to $\{\#3, \#4\} \cap \{\#1, \#2, \#3\} = \{\#3\}$.

3 The PSO Approach

To compute the neighborhood of a new problem, we use the PSO algorithm. The PSO search space is traversed with the objective to minimize a fitness function expressing a semantic of distance to the target problem to resolve. Thus, at the end

of the process, the computed solution constitutes an artificial neighborhood of the target problem. It will be provided as input to the memory access device.

Every source problem computed by the PSO module will be directly pointed to in CBR search space via the net (see Figure 1).

The retrieval step is based on problem description only, but could also include some solution descriptors to compel the retrieval process [5]. We focus now on the PSO module, and propose an approach to construct the target problem neighborhood.

3.1 PSO Background

The roots of the meta heuristic described in this section lie on computing models inspired from ethological studies ([8], [9] and [10]). This inspiration led Kennedy and Eberhart (see [13]) to conceive PSO as a method for function optimization. A PSO algorithm maintains a population of particles ("the swarm"), where each particle represents a location in a multidimensional search space (also called problem space). The particles start at random locations and search for the minimum (or maximum) of a given objective function by moving through the search space. The analogy to reality (in the case of search for a maximum) is that the function measures the quality or amount of the food at each place and the particle swarm searches for the place with the best or most food. The movements of a particle depend only on its velocity and the locations where good solutions have already been found by the particle itself or other (neighboring) particles in the swarm. This is again in analogy to bird flocking where each individual makes its decisions based on cognitive aspects (modeled by the influence of good solutions found by the particle itself) and social aspects (modeled by the influence of good solutions found by other particles). Note that, unlike many deterministic methods for continuous function optimization, PSO uses no gradient information.

More formally, for a swarm of M particles evolving in a N-dimensional search space, the genotype consists of $2N$ parameters, representing the N coordinates of a particle's position, and its N velocity components. A particle moves with an adaptable velocity $\boldsymbol{v}$ that changes according to a linear combination of the difference $\boldsymbol{b}_i(t)$- $\boldsymbol{x}_i(t)$ between the position of the best solution found by the particle up to time t and its current position, and of the difference $\boldsymbol{b}_g(t)$-$\boldsymbol{x}_i(t)$ between the best position ever found by the total population and the particle's current position. Thus, for a particle i, we have

$$\boldsymbol{v}_i(t+1) = w\cdot\boldsymbol{v}_i(t) + c_1\cdot\boldsymbol{U(0,1)}\otimes(\boldsymbol{b}_i(t)-\boldsymbol{x}_i(t)) + c_2\cdot\boldsymbol{U(0,1)}\otimes(\boldsymbol{b}_g(t)-\boldsymbol{x}_i(t) \quad (1)$$

where bold characters denotes vectors, and where ⊗ denotes point–wise vector multiplication, $\boldsymbol{U(0,1)}$ is a function that returns a vector whose positions are randomly generated by a uniform distribution in [0,1], c_1 is the cognitive parameter, c_2 is the social parameter, and w is the inertia factor with range in [0.0,

1.0]. The velocity values must be within a range defined by two parameters v_{min} and v_{max}.

An improvement to the original PSO algorithm is to vary the value of w during execution; starting from a maximal value w_{max}, it is linearly decremented as the number of iterations increases, down to a minimal value w_{min}:

$$w(t) = w_{max} - (w_{max} - w_{min}) \, . \, t/T_{max} \tag{2}$$

In the previous equation, t and T_{max} denote the current iteration and the maximum allowed number of iterations, respectively. The position of each particle at the next step is then computed by summing its current position and its velocity (assuming a unit time step):

$$\boldsymbol{x}_i(t + 1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t + 1) \tag{3}$$

These operations are repeated for T_{max} iterations, or until some other stopping criterion is verified. A typical convergence criterion is the achievement of some desired minimal error with respect to the optimal solution.

The PSO algorithm can be summarized by the following flowchart:

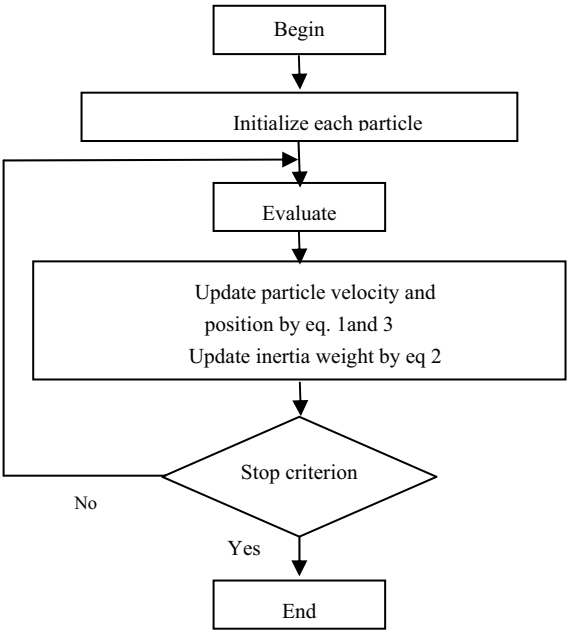


Figure 2 Standard PSO flowchart.

Aside from the basic PSO algorithm, there exist more sophisticated implementations that use different mechanisms for position updating. Among them, two are of interest for this work: confinement and wind dispersion. The confinement mechanism acts by limiting position changes to an interval [11]. It

consists of bounding the position components of a particle in such a way that, for the k^{th} component in the N-dimensional position space, we have:

$$x_{i,k}(t+1) = \text{MIN}(\text{MAX}(x_{i,k}(t) + v_{i,k}(t+1), X_{min}), X_{max}) \quad (3a)$$

where X_{min}=0 and X_{max}=1.

The second mechanism, described in [12] as a chaotic approach, is wind dispersion. An introduction of wind speed and wind direction is considered in order to model the biological atmosphere at the time of updating particle position. The update of the wind speed is given by the following equation:

$$\boldsymbol{vw}(t+1) = \boldsymbol{vw}(t) + v_{op} \cdot \boldsymbol{rand}() + v_{su} \cdot \boldsymbol{rand}() \quad (4)$$

Where $\boldsymbol{vw}$ is the wind velocity, v_{op} is the opposing direction factor equal to -1 and v_{su} is the supporting direction factor equal to 1. The wind speed has one of two effects: particle motion can be opposed or supported by it. The opposing effect slows down the particle in reaching the group's global best solution, whereas the supporting effect increases the particle velocity in reaching in global best solution. Each particle is separately updated by the wind equation. This is supported by the fact that particles are spatially separated from each other, and thus are subject to different dynamic forces from the atmosphere. When the values of the opposing and supporting wind direction, velocities are equal, a static atmosphere is modeled. The position update equation for dimension k in the N-dimensional position space is given by:

$$x_{i,k}(t+1) = x_{i,k}(t) + v_{i,k}(t+1) + vw_k(t+1) \quad (3b)$$

When combining this with confinement, we get:

$$x_{i,k}(t+1) = \text{MIN}(\text{MAX}(x_{i,k}(t) + v_{i,k}(t+1) + vw_k(t+1), X_{min}), X_{max}) \quad (3c)$$

The initial values of wind speed along the wind direction plays an important role in determining the final convergence of the particles to the optimal solution. Also, this parameter ensures the optimal searching of the solution space.

3.2 Proposed Approach

Working with a pure PSO algorithm implies working in a continuous search space. But for real world problems, the problem description space can comprise continuous, discrete and/or nominal attribute values. Hence, we propose to work with two spaces: a search space where particles evolve in a continuous way based on PSO laws [13], and a description space that reflects reality and where entities can have continuous, discrete or nominal values.

Notice that the granularity description in the two spaces is not the same. Indeed, a particle evolves in a continuous way and will be interpreted as the value of a descriptor (or attribute of the description space). On the other hand, the description space is organized as instances, each one being an aggregation of attributes (of miscellaneous natures) with different values. The mapping between the two spaces is ensured by an interpretation mechanism between the continuous values in the first space and the corresponding values in the second. This mechanism consists of:

- Rounding as usually done in discrete PSO (see [14]) when dealing with description space with integer attributes.
- Frequency based selection to handle problems of categorical description. The idea is to interpret the particle position in search space as a frequency corresponding to a categorical value in description space. A frequency table is built during a preprocessing stage by parsing the database of cases and computing the frequency of each attribute value (see Table 2 for an example).Then, the table is used to interpret the values of particle position during the search process.

Thanks to this interpretation mechanism, the PSO algorithm keeps functioning as a continuous model, but position interpretation changes and the result serves to evaluate the fitness function according to the semantics of the description space. Thus, the position, velocity, and inertia factors remain evolving in the continuous search space, and only the fitness function ψ is evaluated with interpreted values corresponding to the mixed attributes in the description space.

Table 2. Partial Frequency table for two descriptors from the Adult database.

Descriptor	Value	Frequency
Race	'Black'	0.0955
	'Amer-Indian-Eskimo'	0.0090
	'Asian-Pac-Islander'	0.0310
	'White'	0.8567
	'Other'	0.0077
Sex	'Male'	0.6706
	'Female'	0.3294

The new PSO algorithm is as follows, where the grayed out boxes indicate the changes to the original algorithm:

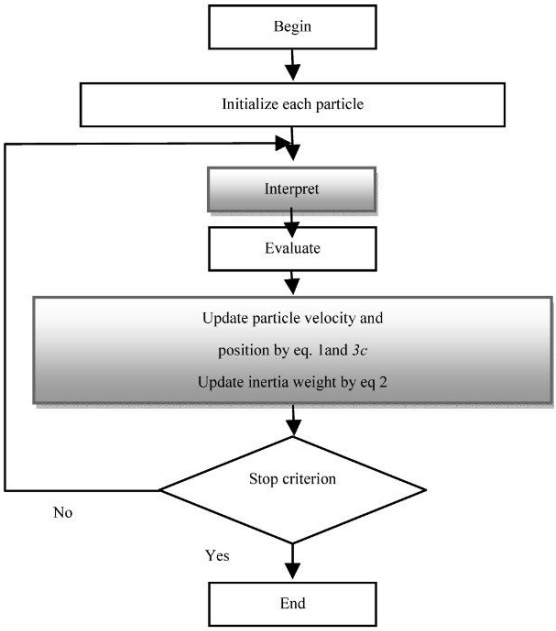


Figure 3 Proposed PSO flowchart.

The fitness function ψ in the context of case retrieval is expressed in term of the distance between the target case and the center of the neighbourhood generated by the PSO algorithm. This distance is to be minimized by the PSO algorithm.

4 Experimental Results and Discussion

In our experiments, we used the Adult dataset, acquired from the UCI Repository [15]. In addition to a high dimensionality characteristic, the Adult dataset has a mixed attribute description since each of its 48842 records contains a mix of continuous and discrete values. The dataset was originally developed for the task of predicting whether a person has an income of over 50K$ per year. Its 14 attributes are either continuous (e.g. age, capital gain and hours of work per week) or categorical (e.g. work class, education and race)

We first constructed a similarity heuristic for the categorical values (see the example in Table 3), in order to allow distance calculation for the fitness function. The heuristic is based on an overlap calculation [16]. The overlap measure simply counts the number of attributes that match in two data instances. The range of per-attribute similarity for the overlap measure is [0; 1], with a value of 0 occurring when there is no match, a value of 1 occurring when the attribute values match and any other value between 0 and 1 when the match is in between. In our study we considered the unknown value '?' as having a 0 match.

Table 3. Example of similarity heuristic for categorical attribute 'race'. The symbol '?' is used for unknown value.

	White	Asian-Pac-Islander	Amer-Indian-Eskimo	Other	Black	?
White	1,00	0,00	0,00	0,25	0,00	0,12
Asian-Pac-Islander	0,00	1,00	0,50	0,25	0,00	0,12
Amer-Indian-Eskimo	0,00	0,50	1,00	0,25	0,00	0,12
Other	0,25	0,25	0,25	1,00	0,25	0,12
Black	0,00	0,00	0,00	0,25	1,00	0,12
?	0,12	0,12	0,12	0,12	0,12	1,00

All the results presented in this section represent the mean of 10 runs executed on a processor of type AMD Athlon™ 64 x 2 Dual-Core processor 3800+, running at a clock frequency of 2.00 GHz, with 2.90 Go of RAM. The PSO parameters were set to v_{max}=0.5, v_{min}=−0.5, c_1=1.7, c_2=1.7, w_{max}=0.9, w_{min}=0.4, the number of particles= 10 and T_{max} = *1000*. Figures 4 to 6 resume the obtained results.

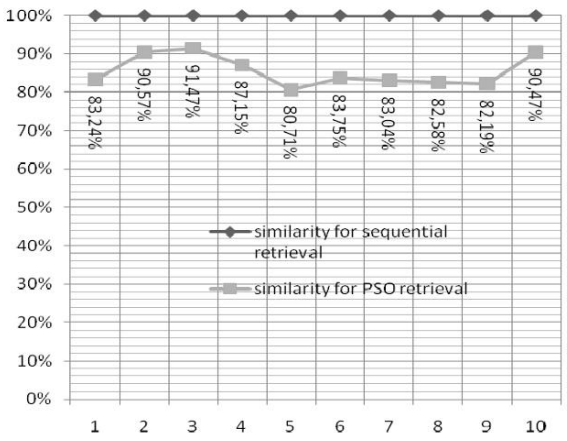


Figure 4 Comparative similarity rate (on y axis) of linear search and PSO search for 1st group of request (on x axis).

We first choose ten requests for which there exists a perfect match (similarity equal to 1) in the case base. As expected, the characteristics of exhaustiveness of the sequential retrieval led to a perfect retrieval of those cases (Figure 4); on the other hand, the PSO recall accuracy varied between 80.71% and 91.47%, depending on the request.

We also compared the time consumption for the two approaches. Figure 5 shows that PSO retrieval reduces this parameter by nearly half in comparison to sequential search.

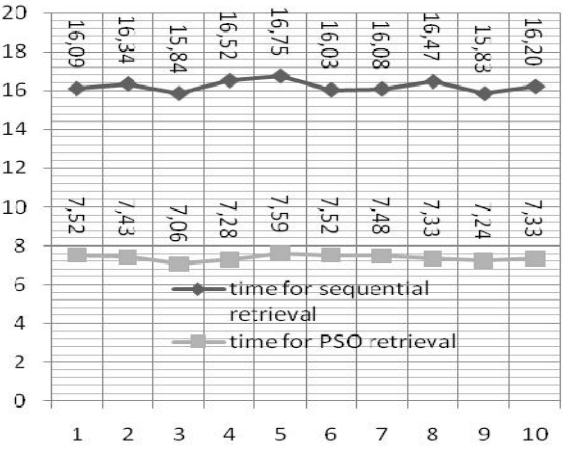


Figure 5 Comparative computation time (in seconds on y axis) of linear search and PSO search for 1st group of request (on x axis).

We then chose ten requests for which there exists a less perfect similarity in the case base (to avoid the perfect retrieval situation). The characteristics of exhaustiveness for the sequential retrieval should also be able to retrieve the best similar cases (Figure 6), but we notice that PSO search performs better for some requests (1, 2 and 6). This could mean that the generated or artificial neighborhood is more similar than the effective neighborhood, so richer than the actual case base. In other terms, the case base covering is not sufficiently rich. The generated neighborhood is a good indicator of case base coverage quality. It could be used for case base enrichment.

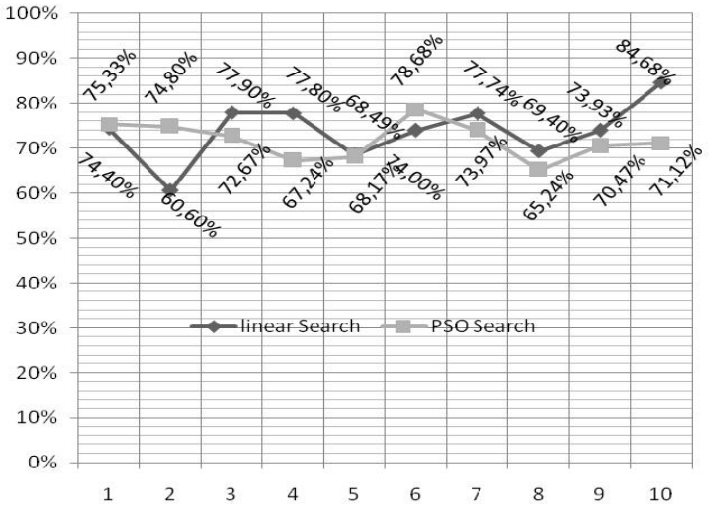


Figure 6 Comparative similarity rate (on y axis) of linear search and PSO search for 2nd group of request (on x axis).

The time for PSO retrieval remains between 6 seconds and 8 seconds, and is again approximately half that of linear search.

5 Conclusion and future work

Many different approaches of case memory models have been proposed in the literature (see section 2.1). By comparison, the PSO approach appears to be interesting for multiple reasons:

- Flexible knowledge representation.
- Good computation performances.
- Suitable for space exploration.
- A large scale of applicability.

Our approach leans on a memory structure that is reachable by content like CRN. In addition, it avoids the inconvenient of classical index-based approaches. It also avoids the need for prior computation and construction of similarity links that add complexity to the construction of memory models like Fish & Shrink and CRN. Finally, it offers flexibility (no pre established links) and easiness of construction with a uniform knowledge representation according to the PSO module. On the other hand, the approach represents a general framework; when considering a specific application field, parameter tuning is required in order to improve the convergence.

The ideal CBR memory is one that simultaneously speeds up the retrieval step while improving the accuracy and robustness of the task performed by the reasoner, particularly the reuse stage, thereby positively influencing the retrieval, reuse, and other steps [17]. As a possible extension, we can consider adding adaptability guided retrieval to the fitness function, like in [18]. Hence, the function to optimize in the PSO approach will express a semantic of reuse combining similarity and copy cost with dissimilarity and adapt cost.

References

1. S.K. Pal, S. C. Shiu, “Foundations of Soft Case-Based Reasoning”, ed. John Wiley & Sons Inc, (2004).
2. Kolodner J., “Case Based Reasoning”, ed. Morgan Kaufmann, (1993).
3. I. Bichindaritz, “Memory Organization As The Missing Link Between Case-Based Reasoning And Information Retrieval In Biomedicine”, Computational Intelligence, Volume 22, Number 3/4, pp: 148-160, Blackwell Publishing , (2006)
4. Bartsch-Spörl B. & al., “Case-Based Reasoning Surveys and Future Direction”, ed Springer, (1999).
5. Lenz M. et al, “Diagnosis and decision support”, in: LNAI 1400, ed. Springer, (1998).
6. Schaaf J. W., “Fish and Shrink. A next step towards efficient case retrieval in large scaled case bases”, in Advances in Case-Based Reasoning, pp: 362-376, Lecture Notes in Computer Science, Ed. Springer Berlin / Heidelberg, (1996).
7. Lenz M., « Case Retrieval Nets as a Model for Building Flexible Information Systems », PhD Dissertation, Humboldt University, Berlin, Germany, (1999).
8. Kennedy,J., Eberhart, R. Swarm Intelligence. Ed. Morgan Kaufmann, (2001).
9. A. Abraham, He Guo, Hongbo Liu, “Swarm Intelligence: Foundations, Perspectives and Applications”, Studies in Computational Intelligence (SCI) 26, 3–25, Springer-Verlag Berlin Heidelberg (2006).
10. Engelbrecht, A., P. Computational Intelligence An Introduction. John Willey & Sons Editions, (2007).
11. Clerc, M. L’optimisation par essaim particulaire: versions paramétriques et adaptatives. Ed. Hermes science publications, Lavoisier, Paris, (2005).
12. Chandramouli, K. and Izquierdo, E., "Image Classification using Chaotic Particle Swarm Optimization," in Proc. International Conference on Image Processing (ICIP '06), (2006).
13. Kennedy J, Eberhart R, Particle swarm optimization. In: Proceedings of the 4th IEEE international conference on neural networks, Perth, Australia, pp 1942–1948, (1995).
14. M. G. H. Omran, A. Engelbrecht, and A. Salman, "Barebones particle swarm for integer programming problems," in Proc. IEEE Swarm Intelligence Symposium, (2007).
15. Asuncion, A. & Newman, D.J.. UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science, (2007).
16. S. Boriah, V. Chandola, V. Kumar, “Similarity Measures for Categorical Data: A Comparative Evaluation”, in Proceedings of SIAM Data Mining Conference, Atlanta, GA, April (2008).
17. I. Bichindaritz, “Memory Structures and Organization in Case-Based Reasoning”, Studies in Computational Intelligence (SCI) 73, 175–194, Springer-Verlag Berlin Heidelberg (2008).
18. N. Nouaouria, M. Boukadoum, “Case Retrieval with Combined Adaptability and Similarity Criteria: Application to Case Retrieval Nets”, In: Proceedings of ICCBR’2010, I. Bichindaritz and S. Montani (Eds.), LNAI 6176, pp. 242–256, (2010).

Dynamic Pricing with Neural Network Demand Models and Evolutionary Algorithms

S. Shakya, M. Kern, G. Owusu[1] and C. M. Chin[2]

Abstract The use of neural networks for demand forecasting has been previously explored in dynamic pricing literatures. However, not much has been done in its use for optimising pricing policies. In this paper, we build a neural network based demand model and show how evolutionary algorithms can be used to optimise the pricing policy based on this model. There are two key benefits of this approach. Use of neural network makes it flexible enough to model range of different demand scenarios occurring within different products and services, and the use of evolutionary algorithm makes it versatile enough to solve very complex models. We also compare the pricing policies found by neural network model to that found by using other widely used demand models. Our results show that proposed model is more consistent, adapts well in a range of different scenarios, and in general, finds more accurate pricing policy than the other three compared models.

1 Introduction

Dynamic pricing [21, 10, 20] is a pricing strategy where a firm adjust the price for their products and services as a function of its perceived demand at different times. In other words, dynamic pricing is to sell the product to the right customers, at the right time, with the right price, in order to maximise the profit. Traditionally, it has been applied in service industries, such as airlines, hotels and rentals [11]. For example, in airlines, the price for a seat changes according to the time remaining for the flight and according to the number of available seats. Recent developments in information technology and eCommerce have led the dynamic pricing to spread over wide range of other industries such as retail [9, 4, 2], wholesale [14] and auction [16].

1 Business Modelling & Operational Transformation Practice, BT Innovate & Design, Ipswich, IP5 3RE, UK. {sid.shakya, mathias.kern, gilbert.owusu }@bt.com

2 Core Design Team, BT Innovate & Design, Ipswich, IP5 3RE, UK. eric.chin@bt.com

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_16,

The key idea in dynamic pricing is to model the effect of interaction between number of different factors, such as price for the product at different times, have on demand, and use that model (known as demand model [21] or price-response model [14]) to optimise the pricing policy. In practice, various assumptions are made on how these factors interact, resulting in different types of demand models. For example, assuming price and demand are linearly dependent to each other a linear demand model can be derived. Similarly, assuming a non-linear dependency a non-linear model can be derived.

In this paper, we purpose to use a neural network [22, 1] based demand model. Unlike traditional demand models, they do not make any pre-assumptions about the relationship between different factors. Rather, they *learn* these relationships from the data itself. Some work on using neural network for dynamic pricing have been previously reported, such as demand forecasting [15] and consumer choice modelling [6]. However, little work has been done in using them for optimising the pricing policies. Also traditionally, numerical techniques such as mathematical programming, have been used for optimising pricing policies, which required gradient information about the objective function. However, a neural network based objective function may not be well defined, and therefore gradient information may not be expressed explicitly. In such scenarios, these traditional techniques may not give a good solution. To overcome this issue, we propose to use evolutionary algorithms (EA) [5] to solve the neural network based dynamic pricing problem. They are population based optimisation technique that uses the concept of natural selection and random variation to *evolve* better solution to the problem.

The paper is organised as follows. Section 2 presents the mathematical model of dynamic pricing and shows how it can be formulated as an optimisation problem. Section 3 describes popular demand models used in practice, and shows how these models are fitted to the data to estimate their parameters. Section 4 describes the detail of the proposed demand model based on neural networks. Section 5 describes how EAs are used to solve the dynamic pricing problems. Section 6 presents the experimental results comparing the pricing policies found by the proposed model to that found by other popular models. Finally, Section 7 concludes the paper by summarising key findings and defining future work.

2 A Mathematical Model of Dynamic Pricing

The model of dynamic pricing presented in this section is adopted from [19]. We use N for number of periods in planning horizon, t for any given period in the planning horizon, Q_t for number of production (sales) at period t, P_t for average price of a product at period t, C_t for cost of selling one extra product at period t, and Π for total profit during the entire planning horizon. The total profit, (Π), earned from a product during the planning horizon can be modelled as

$$\Pi = \sum_{t=1}^{N} (P_t Q_t - C_t Q_t) \tag{1}$$

where, Q_t is the total sales (or the production) of the product (which is equal to, or less than, the demand for the product) in period t, P_tQ_t is the total revenue at period t, and C_tQ_t is the variable cost at t. Next, we define some additional constraints a firm needs to impose when defining its policy

a. Capacity constraints – These are constraints on the number of products that can be produced in a given period, and is defined for all t=1...N as

$$M_t \le Q_t \text{ - Lower bound for the capacity constraint,}$$
$$K_t \ge Q_t \text{ - Upper bound for the capacity constraint,} \tag{2}$$

b. Price constraints – These are the constraints on the selling prices for the product in a given period. They are imposed in order to not overprice or to not lose value for the product, and are defined for all t=1...N as

$$\underline{P}_t \le P_t \text{ - Lower bound for the price cap,}$$
$$\overline{P}_t \ge P_t \text{ - Upper bound for the price cap,} \tag{3}$$

Next, we define the demand model. The most important factors that influence demand for a product are their prices. For example, demand at period t can be higher if price at t is lower, but price at $t+1$ is higher. More precisely, demand for a product in a period depends on the price for that product in that period and also the prices for the product in other periods in the planning horizon. Therefore, we write demand in any period t as the function of prices in all the periods as

$$Q_t = \psi_t(P_1, P_2, ..., P_N) \tag{4}$$

Here, $\psi_t(.)$ is the demand function for period t. Depending on different scenarios, $\psi_t(.)$ can have different functional forms. We discuss this in detail in next section. Substituting Q_t from (4) to (1), we get profit as the function of prices, P_t, (usually variable cost, C_t, is known in advance) which can be written as

$$\Pi = \sum_{t=1}^{N} \left[\psi_t(P_1, P_2, ..., P_N)(P_t - C_t)\right] \tag{5}$$

Therefore, from (2), (3), (4) and (5), the general formulation for the dynamic pricing can be written in terms of optimisation problem as,

$$\max_{P_1, P_2, ..., P_N} \Pi = \sum_{t=1}^{N} \left[\psi_t(P_1, P_2, ..., P_N)(P_t - C_t)\right] \tag{6}$$
$$\text{subject to} \quad M_t \le \psi_t(P_1, P_2, ..., P_N) \le K_t, \quad t = 1, ..., N$$
$$\text{and} \quad \underline{P}_t \le P_t \le \overline{P}_t, \quad t = 1, ..., N$$

Here, the goal is to find the optimal pricing policy, ($P_1, P_2, ..., P_N$), that maximises total profit Π, subject to the constraints in (2) and (3).

3 Model of demand

Depending upon the assumptions made to the demand price relationship, $\psi_t(.)$ can have number of different functional forms. The following are three of the most widely used demand (price) models.

a. Linear model: This model is one of the most popular demand models. They assume that price is linearly dependent on production. This can be written, for all t=1...N as

$$Q_t = \psi_t(P_1, P_2, ..., P_N) = a_t + \sum_{j=1}^{N} b_{jt} P_j \tag{7}$$

where, a_t are the intercept of the linear model representing the customer base (total customers willing to buy the product at period t), and b_{jt} are slopes representing the impact of price at time j have on the demand at time t. Note that, in general, the parameter b_{tt} is negative, since higher price for the product in a period is likely to decrease the demand for that product in that period.

b. Exponential model: Exponential model assume that the relationship between price and demand is exponential and can be written for all t=1...N as

$$Q_t = \psi_t(P_1, P_2, ..., P_N) = e^{a_t + \sum_{j=1}^{N} b_{jt} P_j} \tag{8}$$

Here, a_t and b_{jt} are the parameters similar to the linear model representing the impact of the price on the production.

c. Multinomial-logit model: Multinomial-logit models explicitly model the *consumer's choice*, i.e., explicitly estimate the probability of the consumers choosing to buy the product in a period [21], providing extra information for the practitioners. It can be written as (9), where B is the customer base and b_j are the parameters of the model representing the impact of price at time j on demand.

$$Q_t = \psi_t(P_1, P_2, ..., P_N) = B \frac{e^{-b_t P_t}}{1 + \sum_{j=1}^{N} e^{-b_j P_j}} \tag{9}$$

3.1 Estimating parameters of the demand model

Given the data about past prices for the product and the corresponding sales, a demand model is fitted to the data to estimate the model parameters. These parameters are then used for optimising the pricing policies. For example, in the case of the linear model (7) (or exponential model (8)), a linear regression can be done to fit the model to the historical price-demand data and model parameters, a_t and b_{jt}, can be estimated. Similarly, in case of multinomial-logit model (9), a nonlinear curve fitting algorithm or a maximum likelihood method could be used. The estimated model parameters, together with the prices are then passed as an input to the optimisation problem (6), which is then solved to get the profit maximising pricing policies.

4 Neural network demand models

In this paper, we propose neural network as an alternative approach to modelling demand. Depending upon the number of layers and the ways they are connected to each other, wide range of different neural network topology can be defined. For the purpose of our work, we use a fixed topology network with three layers: an input layer, a hidden layer, and an output layer. Nodes in the input layer define the inputs to the model; nodes in the output layer define the outputs of the model and the nodes in the hidden layer helps to correctly define the relationships between inputs and outputs. We build a set of N neural networks, each modelling the demand price relationship for an individual period[3]. In particular, each neural network represents equation (4), which consist of N input nodes defining prices ($P_1,P_2,...,P_N$) and a single output node defining the production , Q_t. The number of hidden nodes, M, is defined as the half of sum of input and output nodes, $M=(N+1)/2$. Figure 1, shows an example of the implemented structure of the neural network, assuming planning horizon has seven periods. Notice that the *bias nodes* (which have the fixed input value of 1, and help to fit the model to the data more accurately) are added to input layer and the hidden layer.

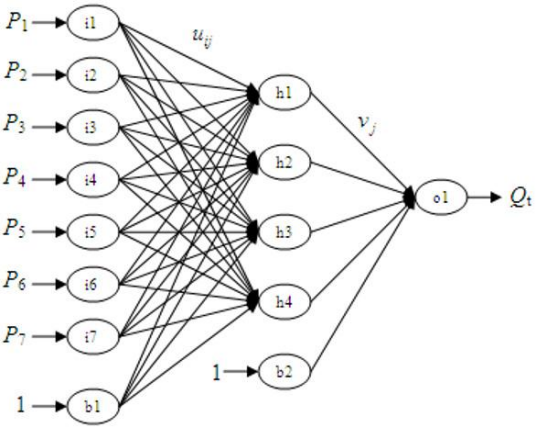


Figure 1, the structure of a neural network demand model with seven inputs and a single output. This models the scenario when there are 7 periods in the planning horizon.

The parameters of the neural network are the weights associated with the connections. We use u_{ij} to represent the weight between input node i and hidden node j. Also, we use u_{bj} to represent the weights between the bias node in input layer and hidden node j. Similarly, we use v_j to represent the weight between

[3] A single neural network with N output could also be defined instead of set of N single networks. However, for simplicity, and also since rest of the model used have N separate functions for each period, we choose to build N single-output neural networks.

hidden node j and the output node. Also, we use v_b to represent the weight between bias node in hidden layer and the output node. Given all the inputs and the weights, we calculate the output of the neural network as

$$Q_t = \alpha\left(\sum_{j=1}^{M}\left(\alpha\left(\sum_{i=1}^{N} P_i u_{ij} + u_{bj}\right)v_j\right) + v_b\right) \tag{10}$$

Here, $\alpha(x)$ is known as the *activation function* of the neural network, which we choose to be of a *sigmoid* form, given by

$$\alpha(x) = \frac{1}{1+e^{-x}} \tag{11}$$

Also, since we use the sigmoid activation function, the output of (10) will be between 0 and 1. Furthermore, it also requires the inputs P_i to be mapped into 0 and 1. We therefore use P_t' and Q_t' to denote P_t and Q_t that are mapped to the values between 0 and 1. Here we do a linear mapping of P_t, which is given by

$$P_t' = mapped(P_t) = P_t^{\min} + \frac{P_t}{P_t^{\max} - P_t^{\min}} \tag{12}$$

where, P_t^{min} is the minimum value which we set to 0 and P_t^{max} is the maximum value which we set to $2\overline{P_t}$. From (10) and (11) demand function for a period t can be written in terms of a neural network with sigmoid activation function as

$$Q_t' = \frac{1}{1+e^{-\left(\sum_{j=1}^{M}\left(\frac{1}{1+e^{-\left(\sum_{i=1}^{N} P_i' u_{ij} + u_{bj}\right)}}v_j\right) + v_b\right)}} \tag{13}$$

Note that the output, Q_t' will be also between 0 and 1 and therefore has to be mapped back to actual Q_t which is obtained using following linear un-mapping procedure

$$Q_t = \psi_t(P_1, P_2, ..., P_N) = unmapped(Q_t') = Q_t^{\min} + Q_t'(Q_t^{\max} - Q_t^{\min}) \tag{14}$$

where, $Q_t^{\min}$ is the minimum value which we set to 0 and $Q_t^{\max}$ is the maximum value which we set to $2K_t$. From (6) and (14), we get formulation of the dynamic pricing as an optimisation problem with neural network demand model as

$$\max_{P_1, P_2, ..., P_N} \Pi = \sum_{t=1}^{N} unmapped(Q_t')(P_t - C_t) \tag{15}$$

subject to $M_t \le unmapped(Q_t') \le K_t \qquad t = 1, ..., N$

and $\underline{P}_t \le P_t \le \overline{P_t} \qquad t = 1, ..., N$

4.1 Estimating parameters of the neural networks

We use a back propagation algorithm to estimate the parameters of the neural networks. Back propagation is a variant of gradient decent algorithm that iteratively improves the weights of the neural network to fit the data. We do not go into further detail on the workflow of the back propagation algorithm, which can be found in [22, 1].

5 An EA approach to dynamic pricing

Typically, an EA starts by randomly generating a population of solutions. In our case the population is a set of pricing policies. Each policy is then evaluated by passing the prices in the policy to the objective function (6) to get the total profit. A subset of good policies (with highest profit value) from the population is then selected and is used to generate new population (known as child population). Different EAs, use different techniques to generate new policies. For example, Genetic algorithms [5], a well know EA, use crossover and mutation approach to generating child population, while, an estimation of distribution algorithms (EDAs) [8] uses a probabilistic approach. The created child population replaces old population and the next set of selection, crossover and mutation operators executes. This process continues until a termination criterion is met.

In this paper we test two EDAs and a GA for solving the neural network based dynamic pricing problem. These algorithms have been previously tested for solving dynamic pricing problem based on linear model (8) [18]. They include Population Based Incremental Learning (PBIL) algorithm [3], Distribution Estimation using Markov Random Field with direct sampling ($DEUM_d$) algorithm [17] and a GA [5]. We also use a non-population based algorithm known as Simulated Annealing (SA) [7] for this problem. For the fairness of the results, we use same algorithms to also solve dynamic pricing problems based on other three demand model, described in Section 3. Due to the lack of space, we do not go into detail on the workflow of these algorithms. Interested readers are referred to [19].

5.1 Constraint handling in EA

The neural network based dynamic pricing problem[4], equation (15), can be seen as a constraint nonlinear optimisation problem. A general constrained optimization problem can be defined as $\max_x f(x), \quad x \in S \subset \Re^n$, subject to the linear or nonlinear constraints $g_i(x) \leq 0, \quad i = 1,...,m$. Here m is the total number of

[4] The dynamic pricing problem based on rest of the model can also be seen as constraint nonlinear optimisation problem and can be solved using this approach.

constraints. One of the most popular ways to solving constrained optimization problems with EAs is by using a penalty function. The idea is to construct a function that penalizes the original objective function for violating the constraints in the model. In order to avoid the penalty, the algorithm tries to focus its search on the feasible part of the search space. Here we use one such technique adopted from [13] and also implemented by [18], and redefine the objective function as

$$F(x) = f(x) - h(k)H(x), \quad x \in S \subset \Re^n, \tag{16}$$

where, *f(x)* is the original objective function (in our case, it is defined by profit, Π, in equation (6). *h(k)H(x)* is the penalising part of the function, where *H(x)* is the main penalty factor (equals to 0 when no constraints are violated) and *h(k)* is known as the dynamically modified penalty value that intensifies the level of penalty according to the algorithm's current iteration *k*. Due to limited space, we do not describe these factors in detail, interested readers are referred to [13] [18].

5.2 Solution representation in EA

A solution, *x*, is represented as a set $P = \{P_1, P_2, ..., P_N\}$, where each P_i is represented by a bit-string of length *l*. The total length of a bit-string solution, $x = \{x_1, x_2, ..., x_n\}$, where $x_i \in \{0,1\}$, is therefore, equal to $n = l \times N$. The goal of an algorithm is to maximize the penalty function defined in (16). We can write the equation to decode the *l* bit to a P_t ranging between $\underline{P_t}$ to $\overline{P_t}$ as

$$P_t = \underline{P_t} + \left[\frac{decoded\_l\_bit\_P_t}{2^l} \times (\overline{P_t} - \underline{P_t}) \right] \tag{17}$$

Where, $decoded\_l\_bit\_P_t$ is the decimal decoded version of the *l* bit representing P_t.

6 Experiments and results

The aim of our experiment is to compare the pricing policy found by neural network based approach to that found by other three demand models. For this purpose a large number of data sets were generated using a number of different source models. All four demand models were then fitted to the data to estimate the model parameters, which were then used for optimisation.

Similar to the observations made in [18], we found that, apart from SA, the three EAs tested in this paper, PBIL, $DEUM_d$ and GA, all performed well in this problem, giving very similar profit values. Since our aim is to compare the performance of optimisation with different demand functions, and not the performance of different EAs, in this paper we only report the results obtained with PBIL. This is because the performance of PBIL was found to be more

consistent on these problems[5]. Once the optimisation problem was solved, we record the pricing policy found by all four models and compare them to the pricing policy found by the original model (the one that was used to generate the original data). The root mean square error (RMSE), given by (18), was used to compare the pricing policy found by the fitted models to that found by the original model. The model that found the solution closest to that found by original model was then chosen as the best model. Let us describe the experimental setups in more detail.

$$RMSE = \sqrt{\frac{\sum_{i \in K} (\exp ected_i - observed_i)^2}{K}} \tag{18}$$

Where, K is the number of data samples.

6.1 Experimental setups

We set the total period in planning horizon $N = 7$. In particular, we assume that the dynamic pricing problem is to correctly price the product for next week by looking at the historical demand price data for past 60 weeks. Obviously, N can also be interpreted as days, months or years. Three different source models: linear, exponential and multinomial-logit, were used to generate data. For each model, we choose three parameter sets, each modelling different scenarios. For each parameter set, we generate five different instances of demand price dataset. Each data set contained 60 records of weekly pricing policy and corresponding sales. The total datasets generated from a model was 5×3=15. Therefore, from three different models, total of 3×15=45 datasets were generated.

Following is the procedure for generating datasets.

1. Randomly generate daily prices for 60 weeks
2. Pass each set of weekly prices through the original demand model and estimate the corresponding production

In order to test the performance of the demand models following steps were performed. First of all the optimisation problem was solved with PBIL using the original model with the original parameter set. The found policy was taken as the best policy for that dataset. PBIL was tuned as follows: each P_t was represented using 12 bit vector, therefore the total length of the bit string solution was 12×7=84, population size was set to 400 and the selection size was set to 100, the learning rate was 0.1. The algorithm was allowed to do 100 iterations.

Next, parameters for each of the four models were estimated by fitting them to each of the 45 data sets. We used linear regression method to fit linear and exponential demand models to the data. Similarly, we used a non linear least

[5] Although, we believe that both GA and $DEUM_d$ would have given us similar conclusions with regard to the demand model performance.

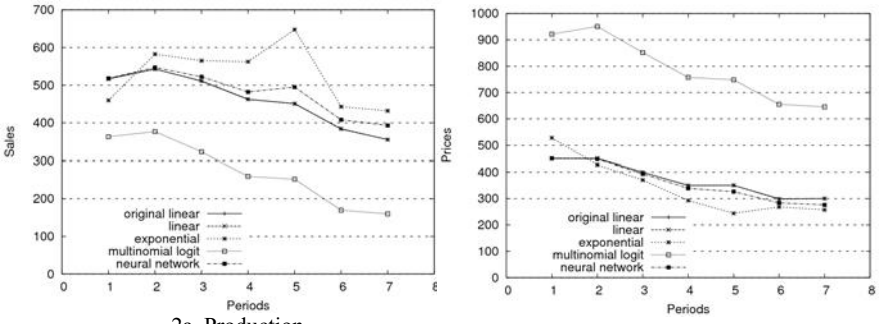
square minimisation method to fit multinomial-logit model. For neural network, we used a back propagation algorithm. Once the parameters were estimated, the optimisation problem was fully specified, which was then solved using PBIL. RMSE between the pricing policies found by these four models and the optimal pricing policy (found using original model) was then computed.

6.2 Results

Table 1 shows the RMSE for each demand model on the entire 45 datasets. Here, the first column defines the type of model with corresponding parameter set (represented by 1, 2 or 3) used to generate the data set; second column defines the type of the demand model that was fitted to the data. Next 5 columns show the RMSE for each of the models on five different instances of the dataset. The final column, gives the average RMSE for each model type on these five instances.

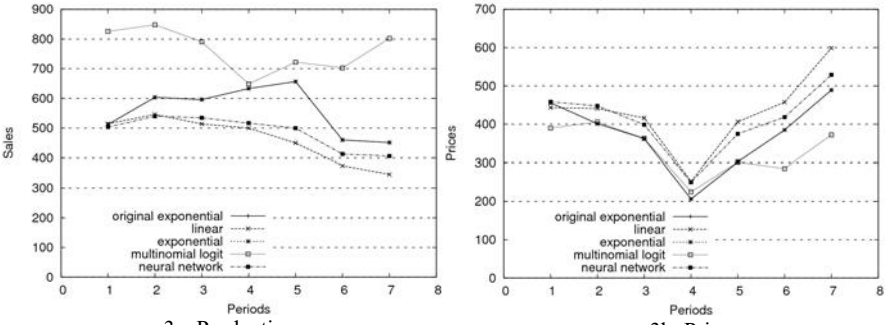
Table 1. RMSE for each demand model on the entire 45 instances

Data	Model	I1	I2	I3	I4	I5	Avg
linear-1	linear	0.76	1.28	1.74	1.48	2.88	1.63
	exponential	80.49	84.28	66.78	78.74	84.50	78.96
	multinomial logit	327.44	329.67	333.58	332.30	321.66	328.93
	Neural network	20.77	18.54	23.73	15.82	25.42	20.85
linear-2	linear	1.44	1.87	1.97	1.56	1.00	1.57
	exponential	222.04	228.17	223.78	227.20	227.20	225.68
	multinomial logit	41.85	42.69	41.97	54.89	65.29	49.34
	Neural network	53.17	38.98	44.93	35.93	31.77	40.96
linear-3	linear	0.95	1.37	2.07	2.00	1.09	1.50
	exponential	93.45	95.13	92.83	92.83	95.72	93.99
	multinomial logit	47.69	34.78	64.00	54.44	87.86	57.75
	Neural network	40.00	26.12	32.52	36.13	31.34	33.22
exp-1	linear	80.27	80.34	80.01	80.27	80.21	80.22
	exponential	9.90	2.18	8.16	5.26	6.17	6.33
	multinomial logit	189.35	188.62	202.71	205.48	192.59	195.75
	Neural network	56.51	55.39	58.30	54.14	54.06	55.68
exp-2	linear	76.75	76.93	75.20	74.39	75.40	75.73
	exponential	6.21	5.82	6.40	5.73	3.38	5.51
	multinomial logit	356.56	355.68	356.68	361.31	359.58	357.96
	Neural network	50.74	48.10	51.67	47.96	49.13	49.52
exp-3	linear	94.06	93.88	92.29	93.62	91.23	93.02
	exponential	0.18	5.71	7.02	5.29	8.43	5.33
	multinomial logit	171.29	165.26	176.20	161.77	161.77	167.26
	Neural network	67.49	63.61	57.66	65.27	65.27	63.86
MNL-1	linear	46.74	47.84	38.75	38.57	40.22	42.42
	exponential	100.24	84.65	89.54	99.74	101.86	95.21
	multinomial logit	35.94	19.86	23.97	37.24	32.42	29.89
	Neural network	44.03	31.58	31.99	37.81	45.00	38.08
MNL-2	linear	67.56	94.81	68.29	81.73	63.48	75.17
	exponential	169.79	175.16	171.81	184.29	185.86	177.38
	multinomial logit	16.87	5.12	10.23	13.90	13.63	11.95
	Neural network	58.89	62.74	67.15	61.87	65.16	63.16
MNL-3	linear	71.84	71.65	72.08	72.84	73.68	72.42
	exponential	53.05	52.40	52.83	53.05	53.35	52.94
	multinomial logit	6.27	0.88	0.43	0.91	0.61	1.82
	Neural network	56.86	52.15	52.04	43.19	47.99	50.44



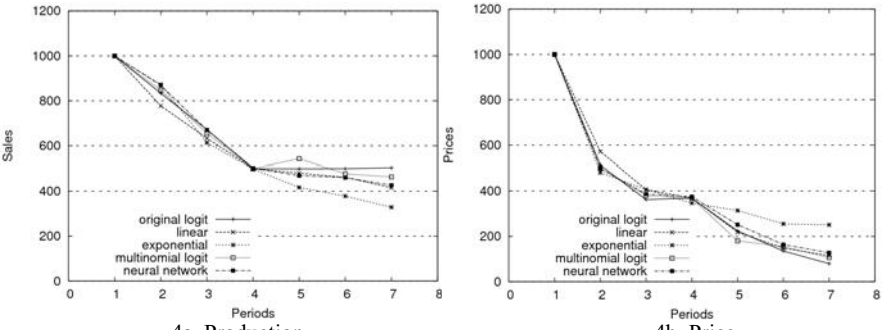
2a, Production 2b, Price

Figure 2, a typical policy found by all four models on linear dataset.



3a, Production 3b, Price

Figure 3, a typical policy found by all four models on exponential dataset.



4a, Production 4b, Price

Figure 4, a typical policy found by all four models on multinomial logit dataset.

We can notice that for the data set sampled from a linear model (we call it linear data set), the linear demand model gives the best performance, i.e the RMSE error is very low[6]. This is expected, since the linearity assumption made by the demand

[6] Although the model is the exact model of the data, the estimated parameters can have some estimation error. This error may depend on the level of noise in the data and the accuracy of the model fitting algorithm used. Consequently, the RMSE of the exact model is not equal to 0.

model exactly fits the data distribution. This result, therefore, is not very interesting. However, the interesting result here is that the neural network had the lowest RMSE amongst the remaining models, i.e. could give the closest-to-optimum results even without making any linearity assumption.

The case is similar with exponential and multinomial-logit datasets. As expected, the demand model that exactly matches the data distribution performs best. The neural network is the best among remaining demand models. Table 2, shows this clearly, where the overall average is presented for each data type.

Table 2. Average RMSE for each models over each of the three data types

Data	Model	Avg-1	Avg-2	Avg-3	Avg-all
Linear	linear	1.63	1.57	1.50	1.56
	exponential	78.96	225.68	93.99	132.88
	multinomial logit	328.93	49.34	57.75	145.34
	Neural network	20.85	40.96	33.22	31.68
Exp	linear	80.22	75.73	93.02	82.99
	exponential	6.33	5.51	5.33	5.72
	multinomial logit	195.75	357.96	167.26	240.32
	Neural network	55.68	49.52	63.86	56.35
MNL	linear	42.42	75.17	72.42	63.34
	exponential	95.21	177.38	52.94	108.51
	multinomial logit	29.89	11.95	1.82	14.55
	Neural network	38.08	63.16	50.44	50.56

Also, in Figures 2a, and 2b, a typical policy found by each of these four models and how they compare to the policy found by original model on linear data set is shown, where Figure 2a is for the production and 2b is for price. Similarly, Figures 3a, 3b shows the same information for exponential data and Figure 4a, 4b shows it for multinomial-logit dataset. Again, these figures show that the policy found by the model matching the source model is closest to the optimal policy. The policy suggested by neural network is closest among the remaining models and produces a similar policy curve as the optimal model. It is important to have the policy-curve similar to the optimal policy, since they verify that the model is correctly representing the scenario and by using such policy the profit will be closer to the optimal policy.

Finally, in table 3, the overall average performance of all the models over all 45 dataset is presented. The key figure to note here is the overall average RMSE for neural network, which is the lowest among all four demand models. This result is particularly important since it shows that, in a real world scenario, where the data distribution may not be available in advance, neural network gives the best estimate of demand price relationship. Subsequently, by using such demand price relationship, the policy found should be closer to the optimal policy. We note that the overall average RMSE for rest of the models are somewhat biased, since they include the very low RMSE obtained by them on data generated using matching source model. However, even when the comparison is made with such result, the performance of neural network is better.

Table 3. Grand average RMSE of all the models over all 45 dataset

Data	Model	Avg-linear	Avg-exp	Avg-MNL	Grand-Avg
All	linear	1.56	82.99	63.34	49.30
	exponential	132.88	5.72	108.51	82.37
	multinomial logit	145.34	240.32	14.55	133.40
	Neural network	31.68	56.35	50.56	**46.20**

This confirms that neural network is the most consistent model for pricing, which can be fitted to the wide range of data. This is in contrast to the other models, which do well when the data source matches the model, but gives poor results when the dataset does not matches the model. This result also suggests that when the data source is not known, neural network is the safest model to use in order to get a reliable pricing policy.

7 Conclusions

In this paper we have shown how we can use a neural network and EAs for optimising pricing policies in dynamic pricing. A number of experiments have been performed comparing pricing policy **o**btained using neural network with that obtained using other popular demand models, which suggest that neural networks are the most consistent models, and gives result closer-to-optimal in range of different scenarios. This is an important result which encourages the dynamic pricing community to implement neural networks as an alternative model for optimising pricing policy. This is particularly important since, in real life scenarios, the data model is most likely to be not known.

It has been shown that correctly choosing the neural network topology can approximate any nonlinear function and can give a more accurate model. We believe that by implementing a topology learning process in current approach, the improved pricing policy could be found. Also, a simple back propagation has been used to fit the neural network to the data. There are, however, a range of other techniques, including the use of EAs, for training neural networks. Using more advanced training methods are likely to give better fit model, again resulting in more accurate pricing policy. Furthermore, it would also be interesting to compare the performance of EA optimised pricing policy with that optimised using other traditional methods, such as mathematical programming. All of these remain the part of the future works.

This work is a continuation to the work presented in [20], where a generic pricing system was described as a component of the Field Optimisation Toolkit [12]. The addition of proposed neural network based demand modelling makes the system more versatile, as it enables automatic learning of the model from the historical data, rather than the manual model building process currently required.

References

1. Arbib, Michael A. (Ed.) (1995). *The Handbook of Brain Theory and Neural Networks*. MIT Press.
2. Baker, W., Marn, M. V., Zawada, C., "Price Smarter on the Net, (2001) " *Harvard Business Review*, Vol. 79, No. 2, February 2001
3. Baluja, S. (1994). *Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning,*. Technical. Report CMU-CS-94-163, Pittsburgh, PA.
4. Ferdows, K., Lewis, M. A., Machura, J. A.D. M. (2004), Rapid-fire fulfilment, *Harvard Business Review,* 82(11), 104-110.
5. Goldberg, D. (1989). Genetic Algorithms in Search, optimization, and Machine Learning. Addison-Wesley.
6. Hruschka H, Fettes W, Probst M (2004) "An empirical comparison of the validity of a neural net based multinomial logit choice model to alternative model specifications." European Journal of Operations Research 159: 166-180.
7. Kirkpatrick, S., Gelatt, C. D. Jr., Vecchi, M. P. (1983). "Optimization by Simulated Annealing", *Science*, 220, 4598, 671-680, 1983
8. Larrañaga, P. and Lozano, J. A. (2001). *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer Academic Publishers.
9. McWilliams, G. (2001). Lean machine: How Dell fine-tunes its pc pricing to gain edge in slow market. *Wall Street Journal*, June 8
10. Narahari, Y., Raju, C. V., Ravikumar K. and Shah, S. (2005) Dynamic pricing models for electronic business, Sadhana, vol. 30, Part 2 & 3, pages 231-256, April/June 2005
11. Netessine, S. and R. Shumsky (2002), "Introduction to the Theory and Practice of Yield Management" *INFORMS Transactions on Education*, Vol. 3, No. 1, http://ite.informs.org/Vol3No1/NetessineShumsky/
12. Owusu, G., Voudouris, C., Kern, M., Garyfalos, A., Anim-Ansah, G., Virginas, B.: On Optimising Resource Planning in BT with FOS. In: Proceedings International Conference on Service Systems and Service Management , pp. 541-546, (2006)
13. Parsopoulos, K.E. and Vrahatis. M.N. Particle swarm optimization method for constrained optimization problems. In P. Sincak, J. Vascak, V. Kvasnicka, and J. Pospichal, editors, Intelligent Technologies--Theory and Application: New Trends in Intelligent Technologies, volume 76 of Frontiers in Artificial Intelligence and Applications, pages 214--220. IOS Press, 2002.
14. Phillips, R.L. (2005): *Pricing and revenue optimization*. Stanford University Press.
15. Qi M, Yang S (2003) "Forecasting consumer credit card adoption: what can we learn about the utility function?" International Journal of Forecasting 19: 71-85.
16. Sahay A. (2007) How to reap higher profits with dynamic pricing, *MIT Sloan management review*, ISSN 1532-9194, 48(4), 53-60
17. Shakya, S., McCall J., and Brown, D. (2005). Using a Markov Network Model in a Univariate EDA: An Emperical Cost-Benefit Analysis. *In proceedings of Genetic and Evolutionary Computation COnference (GECCO 2005)*, Washington, D.C., USA, 2005. ACM.
18. Shakya, S., Oliveira, F., Owusu G. (2007) *An application of GA and EDA to Dynamic Pricing,* In proceedings of Genetic and Evolutionary Computation COnference (GECCO 2007), Pages 585-592, London, UK. 2007, ACM, ISBN 978-1-59593-697-4.
19. Shakya, S., Oliveira, F. and Owusu G. (2008) *Analysing the Effect of Demand Uncertainty in Dynamic Pricing with EAs*. In M. Bramer, F. Coenen, and M. Petridis, editors, Research and Development in Intelligent Systems XXV, proceedings of AI-2008, Cambridge, UK, 2008. Springer-Verlag London.
20. Shakya, S., Chin C. M., and Owusu G. (2010) An AI-Based System for pricing Diverse Products and Services. Knowledge Based Systems, 23(4), pages 357 – 362, ISSN 0950-7051, May 2010, Elsevier.
21. Talluri K.T., van Ryzin, G.J. (2004): *The Theory and Practice of Revenue Management*. Springer, Berlin Heidelberg New York
22. Wasserman, P.D. (1989). *Neural computing theory and practice*. Van Nostrand Reinhold.

Discretisation Does Affect the Performance of Bayesian Networks

Saskia Robben, Marina Velikova, Peter J.F. Lucas and Maurice Samulski

Abstract In this paper, we study the use of Bayesian networks to interpret breast X-ray images in the context of breast-cancer screening. In particular, we investigate the performance of a manually developed Bayesian network under various discretisation schemes to check whether the probabilistic parameters in the initial manual network with continuous features are optimal and correctly reflect the reality. The classification performance was determined using ROC analysis. A few algorithms perform better than the continuous baseline: best was the entropy-based method of Fayyad and Irani, but also simpler algorithms did outperform the continuous baseline. Two simpler methods with only 3 bins per variable gave results similar to the continuous baseline. These results indicate that it is worthwhile to consider discretising continuous data when developing Bayesian networks and support the practical importance of probabilitistic parameters in determining the network's performance.

1 Introduction

Bayesian networks have been successfully used in many domains in which reasoning with uncertainty is of primary importance [14]. Bayesian networks have the virtue of being model based, allowing one to exploit knowledge, available in literature or from experts, to develop the structure of the network as an acyclic directed graph. Without any doubt, the choice of the Bayesian network structure has a major influence on its performance, as whether a set of variables is dependent of another set of variables, possibly given a third set of variables, or not is precisely what can

Saskia Robben, Marina Velikova and Peter J.F. Lucas
Radboud University Nijmegen, Institute for Computing and Information Sciences, The Netherlands, e-mail: saskia@robben.nu, {marinav, peterl}@cs.ru.nl

Maurice Samulski
Radboud University Nijmegen Medical Centre, Department of Radiology, The Netherlands, e-mail: m.samulski@rad.umcn.nl

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_17,

be derived from the graph structure. However, there exists scientific evidence which goes one step further in emphasising the role of graph structure. In a well-cited paper by Pradhan et al. published in 1996 in the AI journal it was experimentally established that the probabilistic parameters of a Bayesian network have only limited effect on its performance; it was concluded that the structure is the single most important factor determining the Bayesian network's performance [18]. In time, this insight has become general wisdom underlying much of Bayesian network modelling. The results of this paper were in particular compelling as they were based on an extensive study of a variety of large, real-world networks. However, as the authors' conclusions and recent research [8] suggest, the problem of the sensitivity of Bayesian networks to imprecision in their parameters is domain-dependent and requires careful investigation.

Our research was started with the manual development of a Bayesian network model for the interpretation of breast X-ray images, usually called mammograms, that is intended to assist radiologists in the computer-aided detection of breast cancer in the context of national breast-cancer screening programmes. Mammogram interpretation is a hard task, and whether a detected anomaly is cancerous or not is inherently uncertain [6]. Besides working as a method for the classification, Bayesian networks may give the radiologist new insights on how certain aspects of a found irregularity contribute to the decision process by computing relevant posterior probabilities. However, after extensive experimentation it was found that the classification performance of the Bayesian network, although carefully designed, was not particularly good. In recent study, we investigated various structures learnt from data as a means to critique the manual network structure [19]. In this paper, we question whether the probabilistic parameters in the initial expert network are optimal and correctly reflect the reality. As most of the variables modelled by the manual Bayesian network were continuous features, they were represented using conditional Gaussian distributions. A limitation of Gaussian distributions is that they are symmetric, which will not allow capturing asymmetries available in the data. In addition to using other continuous probability distributions, that would allow representing asymmetries, in this study discretisation of the continuous data is exploited as another way to fit the probability distribution to the data.

The remainder of the paper is organised as follows. In the next section we review Bayesian networks, the related work, and the main discretisation techniques used in this study. In Section 3, we briefly present the domain of automated mammographic analysis, and the previously developed Bayesian network model with the related features. A description of the data and the experimental set-up followed with the associated the results are also included. Concluding remarks are given in Section 4.

2 Materials and Methods

2.1 Bayesian Networks

A *Bayesian network* $\mathscr{B}$, BN for short, is defined as a pair $\mathscr{B} = (G, P(X))$, where $G = (N, A)$ is an acyclic directed graph with a set of nodes N and arcs $A \subseteq N \times N$, and $P(X)$ is a joint probability distribution of a set of random variables X. The nodes in N correspond 1–1 to the random variables in X. The graph structure represents independence and dependence information, which can be read-off by inspecting whether or not paths between vertices are blocked by other vertices, taking into account the direction of the arcs, which is called d-separation. If U, V, and W are disjoint subsets of X, then U and V are *conditionally independent* given a third set W if $P(U \mid V, W) = P(U \mid W)$. We denote this conditional independence by $U \perp\!\!\!\perp V \mid W$. Independences read off from the associated graph G using d-separation always hold for P, and are, thus, included in the independence relation $\perp\!\!\!\perp$. A dependence between two nodes can also be interpreted as a causality: a cause (parent node) leads to an effect (child node), e.g. the presence of an "abnormal density" in an X-ray image will mostly likely lead to a higher "contrast" value. This provides for a natural interpretation of BNs, which makes them easy to understand even to people with limited understanding of probability theory.

One advantage of a Bayesian network $\mathscr{B}$ is that it provides for a compact representation of the joint probability distribution $P(X)$ by exploiting the independence information represented in the associated graph. This is done by decomposing the joint probability distribution in a product of conditional probability distributions, or CPDs; each random variable has an associated family of CPDs. A CPD describes the conditional probability distributions of a variable given a possible combination of values for the variables associated to the parents of the variable.

BNs can take many forms, from very complicated network structures, providing a detailed and subtle representation of the uncertainties in a domain, to the simpler naïve Bayesian networks, which have been popular as classifiers. For a more detailed recent description of Bayesian networks, the reader is referred to [14].

2.2 Discretisation

Discretisation of data has been studied for more than 20 years as one of the major preprocessing steps in data analysis. Its goal comprises the transformation of continuous variables into a finite number of discrete values, or ranges, to facilitate: (i) the improvement in classification performance, (ii) the induction process of a classifier, or (iii) the interpretability of the models learnt. Next we discuss some studies related to the development and application of discretisation methods to any of these tasks.

2.2.1 Previous Research

In [11], a categorisation of 8 types of discretisation methods is provided such as supervised vs. unsupervised, parametric vs. non-parametric, global vs. local. In the same study, the authors propose a novel discretisation method based on the so-called wrapper approach where the accuracy information from a classifier is taken into account in the discretisation process in order to guide the search for defining the best ranges of all variables simultaneously and ultimately to improve the classification accuracy of the naïve Bayes classifier.

Other studies have applied discretisation techniques to facilitate the induction of a classifier such as the well-known supervised method of Fayyad and Irani, where the class entropy is used to facilitate the induction of better decision trees [9]. The method is briefly reviewed in the next section as one of the main techniques applied in this study. In [2], the authors compare different algorithms for structure learning of BNs in order to build a model for facilitating an emergency hospital service. The study was based on a real dataset where some of the variables were manually discretised based on meaningful context. In [12] the authors investigated the reduction of the variance introduced by various discretisation techniques for decision tree induction. The results demonstrated that this reduction facilitates the interpretability and stability of the models learnt.

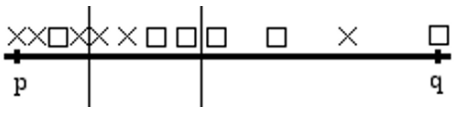
Comparative studies of various discretisation techniques on the performance of naïve Bayes classifiers are provided in [7], [16] and [1], showing improvement in the results compared to the continuous baseline. In addition, in [13] the effectiveness of a number of discretisation methods is evaluated to provide a heuristic for identification of the most appropriate discretisation method to be applied, given the statistical distribution of the attribute to be discretised.

In the current work, discretisation is studied in two different ways, namely with regard to (i) classification performance and (ii) goodness of fit of the resulting probability distribution to the data.

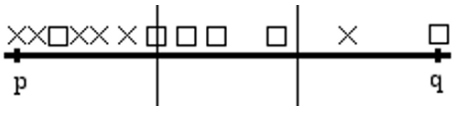
2.2.2 Discretisation Methods

The following methods were investigated and compared to each other:

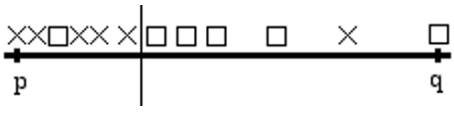
- *Equal Frequency Binning*, or EFB, which determines the bin boundaries by first sorting the data on ascending values and subsequently divides the data in equally sized bins. This algorithm is executed twice: once with 10 bins, for high performance and once with 3 bins, for checking performance while maintaining usability. A visualisation of this method with 3 bins is shown in Figure 1a.
- *Proportional k-Interval (PKI) discretisation*, which is analogous to EFB where the number of bins is equal to the square root of the number of instances [21].
- *Equal Width Binning*, or EWB, which is analogous to EFB but it divides the data in equally ranged bins. The method is applied twice with different number of bins: once with 3 and once with 10 bins. A third variation of the method is used



(a) EFB with 3 bins



(b) EWB with 3 bins



(c) FI

Fig. 1: Visualisation of different discretisation methods on the same (artificial) attribute. The data is first sorted from the lowest (p) to the highest (q) value. The class plays a role only in the FI method in c).

where the number of bins is determined by optimization. A visualisation of this method with 3 bins is shown in Figure 1b.

- The method of *Fayyad and Irani*, or FI for short, which selects a bin boundary based on the minimisation of the class information entropy. The class entropy of a (sub)set S is defined as

$$Ent(S) = -\sum_{i=1}^{k} P(C_i,S)\log P(C_i,S),$$

where $P(C_i,S)$ represents the proportion of instances in S with class C_i and k stands for the number of classes, in our case 2: cancerous and non-cancerous. For each candidate cut point T of an attribute A, a weighted average is calculated of the entropy of the two subsets S_1 and S_2 created by the cut point:

$$E(A,T;S) = |S_1|/|S|Ent(S_1) + |S_2|/|S|Ent(S_2),$$

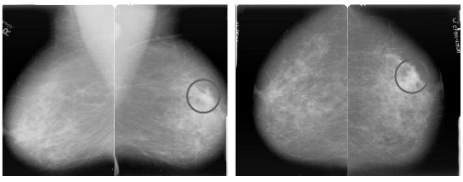
where $|\cdot|$ represents the cardinality of a set. The candidate cut point for which this function is minimal is selected. This process can be repeated on the subclasses to create multiple bins, but the Minimal Description Length (MDL) criterion is used as a stopping criterion to avoid ending up with too many bins. Figure 1c illustrates the result from the algorithm splitting effectively an attribute into two relatively homogeneous bins. For a more detailed explanation the reader is referred to [9].

EFB, PKI and EWB methods are unsupervised as they do not use class information, whereas the FI method is supervised.

3 Discretisation in Automated Mammographic Analysis

3.1 Mammographic Analysis

Mammography is the diagnostic procedure to detect cancer in the breasts using low-dose X-rays. It is currently the most cost-effective procedure used in breast cancer screening programs, which aim at the early detection of breast cancer in asymptomatic women. Every mammographic examination usually consists of two different projections, also called *views*, of the breast: *mediolateral oblique* (MLO) and *craniocaudal* (CC); see Figure 2. The MLO view is a 45° angled side view, showing a part of the pectoral muscles. The CC view is a projection of the breast from above with the nipple centered in the image. The two-view reading is important and can provide more information about the presence of cancer as an abnormality might be obscured in one of the views due to the high breast compression.



(a) MLO (b) CC

Fig. 2: MLO and CC views of both breasts of a patient. A cancerous lesion is marked by the circle

Despite the knowledge and experience of human readers (radiologists), studies have shown that they fail to identify a significant number of cases with breast cancer, mostly due to misinterpretation of the abnormalities observed. To increase the detection rate, *computer-aided detection* (CAD) systems are being developed. These systems use pattern recognition techniques to extract *features* in a mammogram, which are subsequently used to identify *regions* that are possibly suspicious. With such markings, the CAD system can assist radiologists with the mammographic analysis and potentially increase the true detection rate of breast abnormalities.

Currently, the CAD systems analyse every breast view independently and we refer to them as *single-view* CAD systems. While they are capable of detecting an abnormality, they still face problems in its correct classification as cancerous or not. One reason is that these systems fail to account for multi-view dependencies between the breast projections. However, to identify the same lesion on the two projections is not a straightforward task as the two views are projections of a three-dimensional breast. A possible solution is to link all MLO regions with all CC regions of the same breast and then try to identify true and false links. A true link represents a *finding* or *lesion*, and following radiologists practice, we define such a link as the one containing at least one cancerous region.

Previous research has studied the application of Bayesian network technology to model the intrinsic uncertainty of the breast cancer domain [15, 4]. Such models incorporate background knowledge in terms of (causal) relations among variables. However, they use BI-RADS[1] [6] terms to describe a lesion, rather than numerical features automatically extracted from images. This requires the human expert to define and provide the input features a priori, which limits the automatic support of the system. In contrast, the Bayesian network model described in the next section represents the multi-view interpretation principles of the way radiologists analyse mammograms, where the features are automatically extracted from the image.

3.2 Bayesian Network Model

The Bayesian network model used in this study was proposed in [10]; it is reproduced in Figure 3. The BN incorporates MLO and CC features, represented by the white rectangles on the figure, which can be interpreted at the same time, allowing the integration of information from two views. These features are continuous (real-valued) and computed by the single-view CAD system independently per view. Below we describe the most important features, used in the BN model, which determine whether or not two regions linked between both views represent a finding:

- The relative location of the region (`LocX` and `LocY`); some areas of the breast are more likely to contain cancer than others;
- The shortest distance of the region to the skin (`d2skin`);
- High contrast on the mammogram is often associated with a malignancy: tumor tissue absorbs more X-rays than fat and glandular tissue (`Contrast`);
- Indication whether the region margin has a spiky pattern towards the center of a lesion, the so-called "spiculation" (`Spic`); the higher the degree of spiculation, the higher the likelihood for malignancy;
- The presence of a circumscribed lesion, the so-called "focal mass" (`FM`);
- Linear texture (`LinTex`), which is typical for normal breast tissue; the higher the linearity, the lower the likelihood of being cancerous;
- Size of the region (`Size`).
- The malignancy pixel-based likelihood (`DLik`) computed by a neural-network classifier using pixel-based features;
- The false-positive level of a region (`FPLevel`) computed by a neural-network classifier using region-based features; it indicates the average number of normal regions in an image with the same or higher likelihood scores, so the lower its value, the higher the likelihood that the region is cancerous.

The simultaneous interpretation of the MLO and CC features is modelled by the corresponding *hidden* variables (in light grey ovals in the figure), which are not

[1] BI-RADS stands for "Breast Imaging-Reporting and Data System", a quality assurance tool used in mammography.

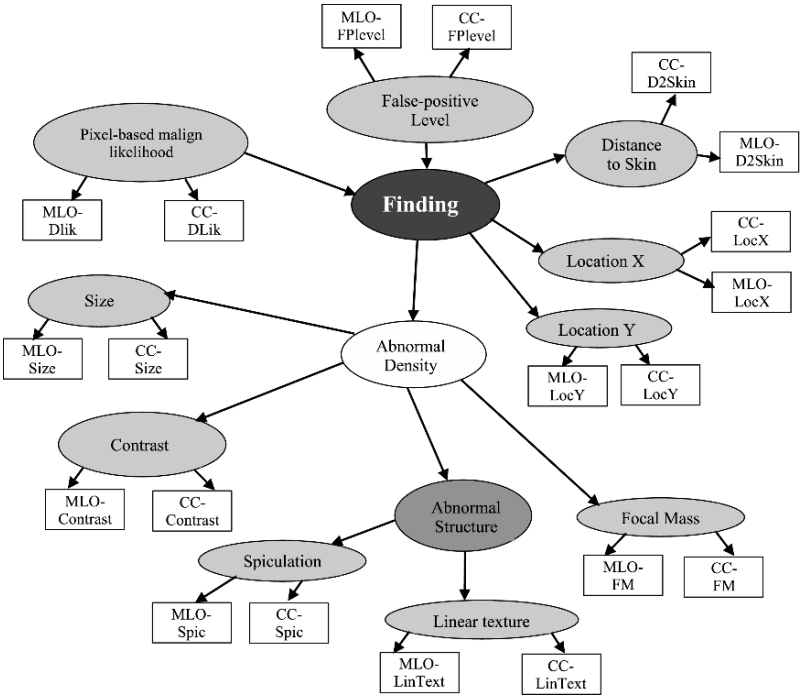


Fig. 3: Bayesian network model for two-view mammographic analysis.

directly observed or measured in the CAD system, but represent the way as radiologists would evaluate the mammographic characteristics of a finding. The variable `Finding` represents the conclusion whether or not there is cancer in the breast, i.e., whether or not two linked regions in MLO and CC views represent a lesion. Central to the BN model are also the hidden variables `Abnormal Density` and `Abnormal Structure`, indicating the presence of abnormal density and structure and they have two states: "present" and "absent".

The causal model was clearly developed for the purpose of two-view mammographic interpretation where the main variable of interest is `Finding`. However, in the screening practice, the ultimate goal is whether or not a patient has cancer and needs to be referred for a further examination. From this perspective, in this study, we analyse the results from the causal model not only at a link level but also at a patient level by taking the maximum out of the probabilities for `Finding` being true, available for the patient's exam.

Despite the clear causal structure and the incorporation of hidden variables capturing human's mammographic interpretation, the model in Figure 3 still performs suboptimal. Various reasons can be attributed such as the set of selected variables, established relationships between the variables and the assumptions regarding the values the variables take. In this work, we explore in depth the last option. In the

original model with continuous variables, the assumption is that all view features can be described by a Gaussian distribution, which in practice is often not the case. Furthermore, the continuous features are difficult to understand and interpret by human readers. Therefore, we conducted an experimental study with mammographic data to see whether the discretisation of MLO and CC features can facilitate the better representation of the distribution of these variables in order to improve the performance and interpretation of the model.

3.3 Data and Experimental Set-up

Data was obtained from the Dutch breast cancer screening practice and includes the mammographic examinations of 795 patients, of which 344 were cancerous. All exams contained both MLO and CC views. All cancerous breasts had one visible lesion in at least one view, which was verified by pathology reports to be malignant (cancerous). Lesion contours were marked by a mammography reader.

For each image (mammogram) we have a number of regions detected by the single-view CAD system. We selected the three most suspicious regions per image (view). Every region is described by continuous features (see Section 3.2), which we further discretised using the methods from Section 2.2, as implemented in the software package WEKA [20]. Based on the ground-truth data, for each region we assign a class value of "cancerous" if the detected region hits a cancerous abnormality and "normal" otherwise. Since a region in one view cannot always be coupled to the corresponding area in the other view due to the the compression and the rotation of the view, for every breast we linked every region from MLO view with every region in the corresponding CC view. For every link we added the class values of "cancerous" ("true") if at least one of the linked regions is cancerous; otherwise the class is "non-cancerous" ("false"). This forms the data for the variable `Finding` in the BN model. We assign analogous classes for the patient based on the ground-truth information. This results in a database where for each breast multiple instances are added, and each instance reflects a link between a CC and a MLO region. The final dataset consists of 14129 links.

To train and evaluate the Bayesian network models with different discretised datasets, we used two-fold cross validation: the dataset is randomly split into two subsets with approximately equal number of observations and proportion of cancerous cases. The data for a whole case belonged to only one of the folds. Each fold is used as a training set and as a test set. We built, trained and tested the networks by using the Bayesian Network Toolbox in Matlab [17]. The learning has been done using the EM algorithm, which is typically used to approximate a probability function given incomplete samples, as the network contains hidden variables [5].

The performance of the BN models learnt with the discretised data was compared with the benchmark model learnt from the continuous data. The comparison analysis is done using the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC), a standard performance measure in the medical im-

age research [3]. We also evaluated the data fitting capabilities of the models learnt on the basis of the log-likelihood measure (*LogLik*):

$$LogLik = \frac{1}{N}\sum_{i=1}^{N} -\log P(C_i|\mathscr{E}_i), \tag{1}$$

where N is the number of exams, C_i and $\mathscr{E}_i$ is the class value and the feature vector of the i-th observation, respectively. Thus, the value of *LogLik* indicates how close the posterior probability distribution is to reality: when $P(C_i|\mathscr{E}_i) = 1$ then $\log P(C_i|\mathscr{E}_i) = 0$ (no extra information); otherwise $-\log P(C_i|\mathscr{E}_i) > 0$.

3.4 Results

Table 1 presents the AUC and log-likelihood test results at a link and patient level. In terms of accuracy at a link level, the FI method performs best, followed by the EFB method with 10 bins. The EWB method with 10 bins also slightly improves upon the baseline approach whereas for the remaining discretisation techniques the performance gets worse. The AUC measure of the PKI method indicates that its performance is close to random classification.

Although the results at a link level are an indicator for the model performance, from a clinical point it is interesting to consider the results at the patient level. The FI method again achieves the best discrimination between cancerous and normal cases, followed by EFB and EWB with 10 bins. EWB with an optimized number of bins now slightly improves compared to the continuous baseline, instead of deteriorating when considered at a link level, whereas EFB with 3 bins shows the opposite pattern. EWB with 3 bins and PKI also show no improvement.

Table 1: AUC and log-likelihood test results obtained from the continuous baseline and the discretisation methods

Method	AUC		*LogLik*	
	Link	Patient	Link	Patient
Continuous variables (baseline)	0.7065	0.6276	0.4658	0.7604
FI	**0.7898**	**0.7548**	0.3819	0.6325
EFB-10	**0.7539**	**0.7331**	0.3943	0.6166
EFB-3	0.7090	0.6113	0.4103	0.6694
EWB-10	**0.7196**	**0.6543**	∞	0.6610
EWB-Optim	0.7041	0.6322	0.4112	0.6722
EWB-3	0.6775	0.5721	0.4085	0.6862
PKI	0.5922	0.5329	0.6702	1.0184

To obtain better insight into the improvement of the classification performance, we plotted the ROC curves for the best performing methods at both link and patient level, as shown in Figures 4 and 5.

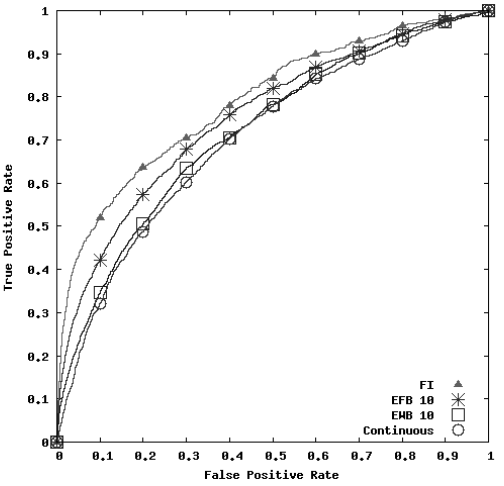


Fig. 4: ROC curves for the best performing discretisation methods against the continuous baseline at a *link* level.

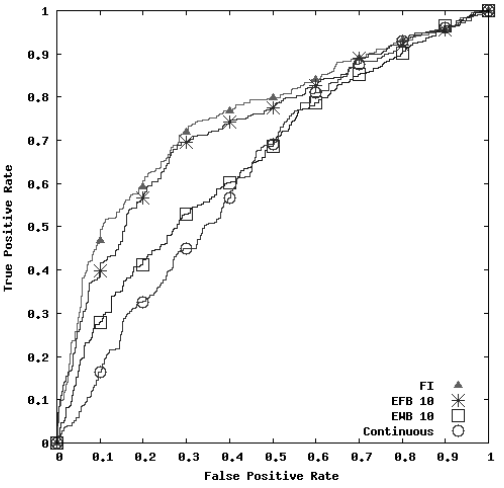


Fig. 5: ROC curves for the best performing discretisation methods against the continuous baseline at a *patient* level.

It is interesting to observe that for the supervised FI method the bigger improvement in the model's performance is in the lower FP range (< 0.5)–a desired result in the screening practice where the number of normal cases is considerably larger than those of the cancerous ones. Furthermore, we note that the curves (and respective AUCs) for all methods are lower at a patient level than those at a link level, as for

the former the number of false positives is much smaller leading to a bigger penalty for a misclassified cancerous case.

We further evaluated the data fitting capabilities of the models with the discrete and continuous data using the log-likelihood measures reported in Table 1. Clearly the FI method fits best to the data at a link level as it achieves the lowest *LogLik* value, and it is followed by EFB with 10 bins. At a patient level, the performance of both methods has the opposite pattern. The EWB method with 10 bins has a *LogLik* value of infinity at a link level because for 6 cancerous links it predicts that the probability for true finding is zero. The BN model with continuous features fits considerably worse to the data in comparison to all models with discretised data except for PKI, indicating mismatch between the model and the data. These results confirm our expectation that discretisation can facilitate the knowledge representation and modelling of the problem of automated mammographic analysis. The PKI method performs worst of all in this study, yielding a considerable drop in classification performance and data fitting capabilities. One reason might be the large number of bins created (over 80). Another explanation could be that the number of bins created by PKI depends on the total amount of instances, that might work when class occurrences are similar, but in our dataset false links are overrepresented.

Finally, Figure 6 illustrates the behaviour of the BN model with discrete MLO and CC features obtained from the FI method for one cancerous link (true finding) from the data. The evidence has been set on the observable nodes, and thereafter the posterior probability of the finding being cancerous has been updated. The model clearly succeeds in the correct classification of the link. Furthermore this model is easier to work and interpret in comparison to the model with continuous features as the former represents closer the knowledge of the human readers.

4 Conclusions

In this paper we investigated the impact of data discretisation on the performance of a manually developed Bayesian network model for automated mammographic analysis. The decision whether or not to discretise data before the modelling step is not straightforward and highly depends on the nature of the data and the problem at hand. As mentioned in the introduction, based on a paper by Pradhan et al. the general wisdom in the field is that the probabilistic parameters are only of secondary importance [18]. However, our research results show otherwise.

The results of the experiments confirmed our expectation that discretisation can improve the representation and the accuracy of the models in comparison to the model with continuous variables. First, the discrete data better capture the conceptual aspects of the way radiologists analyse mammograms and evaluate abnormalities. This allows for easy interpretation and usability of the Bayesian network model. Second, appropriate discretisation provides better approximation of the true probability distribution of the data used and avoids the strong Gaussian assumption

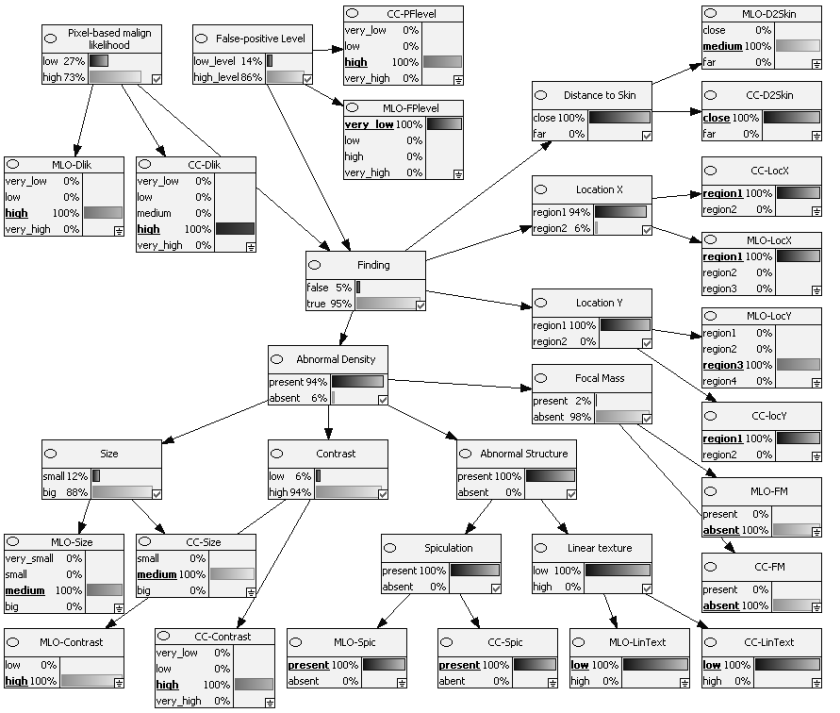


Fig. 6: Bayesian network with evidence set (represented by bold and underlined names of the states) for one cancerous link and posterior probabilities with discretised data using the FI method.

imposed for the continuous variables. As a result the accuracy and the data fitting capabilities of the models improved as shown in this study.

The best performance was achieved by the supervised method of Fayyad and Irani. This is logical as the bin boundaries are more likely to lie on the class boundaries and therefore there is less noise in each of the bins. On the other hand, binning methods such as EWB may create very sparse bins, even empty ones, or may split data with the same characteristics into multiple bins. These problems are common for the unsupervised methods. Nevertheless, as our experiments showed, even the simple binning techniques are capable of improving the classification performance and the data fitting capabilities of the expert Bayesian network model in comparison to the continuous baseline model. Overall, the current study supports the practical importance of probabilitistic parameters in determining the network's performance, especially for complex domains such as medical image interpretation.

A direction for future research is to better optimise the parameters of discretisation. One possibility is to replace the current univariate approach with multivariate discretisation, where the relationships between the variables–typically occurring in practice–are taken into account. Finally, we might also consider the classification accuracy as a guide for the discretisation process.

References

1. Abraham, R., Simha, J.B., Iyengar, S.S.: A comparative analysis of discretization methods for medical datamining with naïve Bayesian classifier. In: Proc. of the Ninth International Conference on Information Technology, pp. 235–236 (2006)
2. Acid, S., de Campos, L.M., Fernandez-Luna, J.M., Rodriguez, S., Rodriguez, J.M., Salcedo, J.L.: A comparison of learning algorithms for Bayesian networks: a case study based on data from an emergency medical service. Artif. Intel. in Medicine **30**(3), 215–232 (2004)
3. Bradley, A.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition **30**(7), 1145–1159 (1997)
4. Burnside, E., Davis, J., Chhatwal, J., Alagoz, O., Lindstrom, M., Geller, B., Littenberg, B., Shaffer, K., Kahn Jr, C., Page, C.: Probabilistic computer model developed from clinical data in national mammography database format to classify mammographic findings. Radiology **251**(3), 663–672 (2009)
5. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B **39**(1), 1–38 (1977)
6. D'Orsi, C., Bassett, L., Berg, W.e.a.: Breast Imaging Reporting and Data System: ACR BI-RADS-Mammography (ed 4) (2003)
7. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: Proc. of the 12th ICML, pp. 194–202 (1995)
8. Druzdzel, M.J., Onisko, A.: Are Bayesian networks sensitive to precision of their parameters? In: Proc. of the International IIS08 Conference, Intelligent Information Systems XVI, pp. 35–44 (2008)
9. Fayyad, U., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proc. of the 13th IJCAI, pp. 1022–1027 (1993)
10. Ferreira, N., Velikova, M., Lucas, P.: Bayesian modelling of multi-view mammography. In: Proc. of the ICML Workshop on Machine Learning for Health-Care Applications (2008)
11. Flores, J.L., Inza, I., naga, P.L.: Wrapper discretization by means of estimation of distribution algorithms. Intelligent Data Analysis **11**(5), 525–545 (2007)
12. Geurts, P., Wehenkel, L.: Investigation and reduction of discretization variance in decision tree induction. Lecture Notes In Computer Science **1810**, 162–170 (2000)
13. Ismail, M.K., Ciesielski, V.: An empirical investigation of the impact of discretization on common data distributions. In: Proc. of the Third Int. Conf. on Hybrid Intelligent Systems: Design and Application of Hybrid Intelligent Systems, pp. 692–701 (2003)
14. Jensen, F., Nielsen, T.: Bayesian networks and decision graphs. Springer Verlag (2007)
15. Kahn, C., Roberts, L., Shaffer, K., Haddawy, P.: Construction of a Bayesian network for mammographic diagnosis of breast cancer. Comp. in Biol. and Medic. **27**(1), 19–29 (1997)
16. Mizianty, M., Kurgan, L., Ogiela, M.: Comparative analysis of the impact of discretization on the classification with naïve Bayes and semi-naïve Bayes classifiers. In: Proc. of the Seventh International Conference on Machine Learning and Applications, pp. 823–828 (2008)
17. Murphy, K.: Bayesian network toolbox (BNT) (2007). http://people.cs.ubc.ca/∼murphyk/Software/BNT/bnt.html
18. Pradhan, A., Henrion, M., Provan, G., del Favero, B., Huang, K.: The sensitivity of belief networks to imprecise probabilities: an experimental investigation. Artificial Intelligence **84**(1-2), 357–357 (1996)
19. Radstake, N., Lucas, P.J.F., Velikova, M., Samulski, M.: Critiquing knowledge representation in medical image interpretation using structure learning. In: Proc. of the Second Workshop "Knowledge Representation for Health Care", Lisbon, Portugal (2010)
20. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, Second Edition. Morgan Kaufmann, San Francisco, CA, USA (2005)
21. Yang, Y., Webb, G.: Proportional k-interval discretization for naïve-Bayes classifiers. In: Machine Learning: ECML 2001, pp. 564–575. Springer (2001)

A Structural Approach to Sensor Placement based on Symbolic Compilation of the Model

Gianluca Torta and Pietro Torasso

Abstract In the present paper we address the problem of computing the Minimal Additional Sensor Sets (MASS) that guarantee a desired level of diagnostic discrimination for a system.
The main contribution of this paper is the extension and the adaptation of techniques based on the symbolic compilation of qualitative system models to a structural approach suitable for the computation of MASS for component-oriented models consisting of sets of numerical equations. In this respect, the paper can be viewed as a bridge across the AI approaches to model-based sensor placement and the Fault Detection and Isolation approaches developed by the Automatic Control community.
We show that the resulting method exploits the symbolic compilation techniques not only as a way to provide computational savings (including some theoretical guarantees on the computational complexity), but it also exhibits interesting new features, most notably the handling of multiple faults.

1 Introduction

In recent years the Model-Based Diagnosis community has devoted a significant amount of attention to the problem of diagnosability and to the related problem of determining a set of sensors that guarantees the diagnosability of a given system (or at least a *desired* level of diagnosability since in some cases it is very hard to assure diagnosability in all possible conditions of the system).

In most cases one is interested in finding a sensor set that is *minimal* according to some criteria such as set inclusion, cardinality or total cost. The search space for the minimal sensor sets is usually specified by the system modeler as a set of potential measurement points, i.e. physical quantities that could be measured by

Gianluca Torta and Pietro Torasso
Dipartimento di Informatica, Università di Torino, C.so Svizzera 125, 10149 Torino (Italy), e-mail: {torta,torasso}@di.unito.it

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_18,

placing a suitable sensor. Such a search space could be constrained either by positive information (e.g. we already have some sensors in place that "come for free") or negative information (e.g. the sensors placed in certain places are too unreliable or prone to failure or too costly). When such constraints are present, the problem of computing the Minimal Sensor Sets (MSS) is transformed into the problem of computing the Minimal Additional Sensor Sets (MASS).

The problem of computing the MSS (or MASS) has been widely studied in the FDI literature (e.g. [6], [5], [3], [4]). It is worth mentioning that the problem is computationally hard and usually requires a significant amount of search in order to find an optimal solution. For this reason, in many cases the problem has been addressed by putting some additional constraints such as the single fault assumption, so that the requirement of diagnosability is simplified by requiring that, for each component, it is possible to discriminate whether it is ok or faulty under the assumption that at most one fault is present in the system.

Recently, in [7], the computation of MASS has been approached in a quite different way by exploiting symbolic representation and compilation techniques. Such a proposal has been developed by taking an AI approach to the problem, and addresses component based systems where the models of the system components are given in terms of qualitative relations (in particular, this approach has been demonstrated on combinatorial digital circuits and qualitative models of hydraulic systems).

The main goal of the present paper is to describe how the techniques proposed in [7] can be extended and adapted in order to deal with system models given in terms of numeric equations.

As mentioned above, the problem of MASS computation for such a class of systems has been previously investigated and many FDI approaches have been proposed. In this paper we concentrate specifically on the approach presented in [6]. Our focus is not only to reproduce, with a novel method, the results of the existing approach, but also to show the benefits of our method in terms of computational costs and flexibility, most notably the release of the single fault assumption, which can be considered too strong in many real world applications.

The application of the proposed method is illustrated using as a test bed a gas turbine subsystem taken from [6].

2 Discriminability and MASS for Qualitative Relational Models

In this section we provide the formal setting for characterizing the notion of diagnostic discriminability and the one of Minimal Additional Sensor Set.

We start from the definition of *System Description* according to which the model is given in terms of discrete variables and qualitative relations among them. In particular, a *System Description* is a pair $SD = (SV, DT)$ where:

- SV is the set of discrete system variables partitioned in C (system components), X (exogenous variables) and E (endogenous variables). We will denote with $D(v)$ the *finite* domain of variable $v \in SV$; in particular, for each $c \in C$, $D(c)$ consists of the

values *ok* and *ab* for representing respectively the nominal and faulty behavioral modes [1] .

- *DT* (Domain Theory) is a relation over the variables in *SV* relating the value of the exogenous variables and of the behavioral modes of the components with the value of the endogenous variables.

In Model Based Diagnosis, it is assumed that the values of exogenous variables X (i.e. external inputs and commands) are known and the values of the C variables (i.e. health states of components) are not known, since they must be determined through diagnosis. As for endogenous variables E, typically some of them are observed with sensors, while others are not.

Computing a MASS consists in determining a (minimal) set of sensors for observing endogenous variables. We denote with S a set of sensors; each sensor $s_e \in S$ is associated with an endogenous variable $e \in E$. The observed endogenous variables will be the ones with an associated sensor; therefore, we denote with $O(S)$ (*system observability*) the set of endogenous variables $e \in E$ s.t. $s_e \in S$.

We are now ready for introducing the notion of discriminability of a component. Such a notion clearly depends on the degree of system observability.

Definition 1. Let S be a set of sensors and $c \in C$ be a system component. We say that c is *discriminable* w.r.t. observability $O(S)$ iff for each instance x of X:

$$\Pi_{O(S)}(\sigma_{c(ok)\wedge x}(DT)) \cap \Pi_{O(S)}(\sigma_{c(ab)\wedge x}(DT)) = \emptyset$$

where Π, σ are the classic *project* and *select* operations defined in the relational algebra.

What the above definition says, is that component c is considered discriminable w.r.t. a given observability $O(S)$ when, for each possible assignment x to the inputs, it is possible to tell whether c is *ok* or *ab* by looking just at the values of the $O(S)$ variables. For diagnosis purpose, this means that we are always able to determine the health status of c from the readings of the sensors S regardless of the inputs x and the health status of the other components $C\backslash\{c\}$.

The latter point has a significant consequence on the generality of the definition: in fact there is no assumption on the maximum number of simultaneous faults that may affect the system, i.e. any combination of *ok* and *ab* values for the components $C\backslash\{c\}$ is allowed.

In this paper we are interested in verifying whether a set of components is discriminable and what kind of observability guarantees such a discrimination. For this reason we introduce the notions of *discriminability requirement* and Minimal Additional Sensor Set (MASS).

Definition 2. A discriminability requirement $\delta = c$ involving a component variable c is satisfied by an observability $O(S)$ iff c is discriminable w.r.t. $O(S)$ according to Definition 1. We denote as $\Delta = \{c_{\Delta,1}, \ldots, c_{\Delta,k}\}$ a set of discriminability requirements; Δ is satisfied by $O(S)$ iff $O(S)$ satisfies the discriminability requirements for $c_{\Delta,1}, \ldots, c_{\Delta,k}$.

[1] Actually, the domain of component variables can contain more than one faulty behavioral mode. For the purpose of this paper, however, it is sufficient to consider the *ab* mode

Definition 3. Given a set of sensors FS already placed on the system, a *Minimal Additional Sensor Set* w.r.t. the set of requirements Δ is a sensor set S^* such that: $FS \cap S^* = \emptyset$; $O(FS) \cup O(S^*)$ satisfies Δ; and for no other sensor set S', $|S'| < |S^*|$, $O(FS) \cup O(S')$ satisfies Δ.

Note that a Minimal Sensor Set is just a MASS where $FS = \emptyset$. The preference criterion for selecting MASS is based on the minimum cardinality, reflecting the assumption that all the sensors have an equal cost; we will comment on sensors with (qualitatively) different costs in section 7.

3 Background Information on Numerical Models

In this section we provide some background information on the class of system models that we address in our work. In particular, a system model is characterized by a set of components and a set of numeric equations, where each equation is possibly associated with one or more components.

The equations are defined in terms of endogenous variables (which are initially assumed to be all unknown) and exogenous variables (which are assumed to be known), and each component can be either in the *ok* or *ab* (abnormal) modes. The upper part of Figure 1 shows the model of a Gas Fuel Subsystem (GFS) as presented in [6], where the names of exogenous variables end with the star character; more details about the GFS model and the involved parameters can be found in [6] and its bibliographic references to the *TIGER* project.

As discussed in [6], from this kind of numeric equation models it is possible to derive a Resolution Process Graph (RPG) which represents a specific causal ordering for the resolution of the equations in the system model (in general, more than one causal ordering is possible). More specifically, the RPG defines the dependency paths among variables which indicate the order in which every equation should be used to determine the values of the unknown variables[2].

In particular, for each unknown variable v, the RPG contains a node N_v with an incoming arc from a node N_{eq} which represents the equation eq matched with v in the RPG. If equation eq involves v plus other endogenous and exogenous variables $v_1, \ldots, v_k$, then nodes $N_{v_1}, \ldots, N_{v_k}$ in the RPG are connected to node N_{eq} with outgoing arcs; the intended meaning is that values $v_1, \ldots, v_k$ will be used for determining the value of v via eq.

The number of equations in the model is, in general, greater or equal to the number of unknown variables; in the latter case some equations, called Redundant Relations (RR), appear as sink nodes in the RPG (i.e. they have only incoming arcs). The lower part of Figure 1 shows the RPG for the GFS model; note that equation $r5$ is a Redundant Relation. For example, variable p_3 has an incoming arc from its matched equation r_1 which, in turn, has incoming arcs from exogenous variable cpd^* and q_3.

[2] The RPG can be obtained straightforwardly from a *perfect matching* between the endogenous variables and a subset of the equations [2].

name	component(s)	equation
r1	Injt	$q_3 = K_{inj} \times \sqrt{p_3 - cpd^*}$
r2		$q_4 - K_{li} \times q_3 = 0$
r3	GCVh	$q_2 = fsg \times \sqrt{fpg_2 - p_3}$
r4	GCVh	$q_3 - KI \times q_2 = 0$
r5	SRVh	$fqg = fsgr \times \sqrt{p_1^* - fpg_2}$
r6	SRVh	$q_2 - KI' \times fqg = 0$
r7	GCVm	$fsg = f(fag, 96hql^*)$
r8	SRVm	$fsgr = f(fagr, 96hql^*)$
r9	GCVm	$fsg = f(fsrout^*, 96hql^*)$
r10	SRVm	$fsgr = f(fprgout^*, fpg_2, 96hql^*)$
r11	SRVm & SRVh	$fpg_2 = f(fprgout^*)$

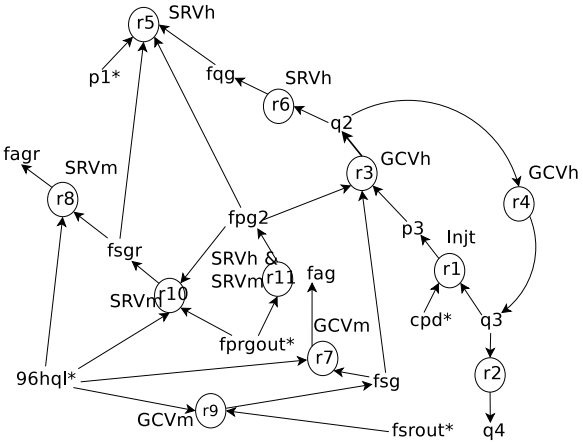


Fig. 1 Model and RPG for the GFS System.

Since RRs are not needed to determine the values of the unknown variables, they can be reformulated in terms of known variables (Analytical Redundant Relations); therefore, it is possible to check whether an RR r is satisfied or not by just looking at the known variables. This is reflected in the value of a variable called *residual* which assumes the value z (for *zero*) if r is satisfied and nz (for *non-zero*) is r is not satisfied.

Since the residual variables can take only two values, they can be modeled as discrete qualitative variables; moreover their value, which depends only on known values, is always known.

As in many other works (including [6]), we assume that the exoneration working hypothesis holds: a faulty component c always implies that the residuals of all the (Analytical) RRs in which it is involved are non-zero.

In our example RPG of Figure 1, all of the system components are involved by the RR $r5$ since, in order to reformulate $r5$ in terms of the (known) exogenous variables $p_1^*, 96hql^*, fprgout^*, fsrout^*$ and cpd^*, we need equations $r1$, $r3$, $r4$, $r6$, $r9$, $r10$, $r11$ (and, of course $r5$ itself) which, all together, involve components $SRVh$, $Injt$, $GCVh$, $GCVm$, $SRVm$, i.e. all of the system components in the GFS.

4 Building a Qualitative Relational Model

In this section we show how, starting from an RPG, it is possible to automatically build a qualitative relational model (i.e. a System Description) extended with special variables for capturing the different levels of observability.

This process is not trivial for two reasons. First of all, it must encode the RPG into a suitable set of qualitative relations. Moreover, different levels of observability must be encoded by *hypothetical sensors* which represent within the system model the sets of sensors that may be placed on the system; more specifically, hypothetical sensors are associated with endogenous variables e which are matched with non-redundant relations r in the RPG (note that the addition of a sensor for observing e makes relation r become a redundant relation, with its own associated residual).

As we will see, the residuals of the RRs in the original RPG and the residuals of new RRs that arise as a consequence of observing endogenous system variables constitute the observable variables in the qualitative relational model.

The adoption of special variables for representing different levels of observability has been pioneered in [7]; however the mechanism for building the extended Domain Theory is quite different (and significantly simpler) from the one described in this paper, since in [7] the original Domain Theory is already expressed in terms of qualitative relations and the special variables (called *switches*) are directly associated with system observables.

We start by defining some of the qualitative system variables of the System Description which correspond to variables in the RPG:

- the set C (system components) which contains a variable c for each component in the RPG with possible values *ok* and *ab* for representing respectively the nominal and faulty behavioral modes
- the set X (exogenous variables) which represent the exogenous inputs and commands; for each $x \in X$, we assume that the actual numeric value of x is always known and therefore the qualitative variable x can only take the value *kwn*
- the set E (endogenous variables) which contains a variable e for each endogenous variable in the RPG with possible values *nom* (e has a nominal value), *abk* (e has an abnormal value which is known, see also the $\hat{E}$ variables below) and *unk* (e has an abnormal value which is unknown)

We also define three additional sets of variables:

- the set HS of hypothetical sensor variables; for each non-RR r in the RPG, there is a variable $hs_{e_r} \in HS$ with possible values *yes* and *no*, whose meaning is that the endogenous variable e_r matched with r in the RPG *is* (resp. *is not*) sensorized
- the set $\hat{E}$ of *propagated* endogenous variables; for each endogenous variable e in the RPG, there is a variable $\hat{e} \in \hat{E}$ with possible values *nom*, *abk* and *unk*. As we will see, in case an endogenous variable e has been observed with a sensor, the value that is propagated to solve the dependent equations may be different than the value of e itself (in particular, when the value of e is *unk*, the qualitative value of $\hat{e}$ may be *abk*)

Algorithm **BuildQR**
builds a qualitative relation QR_r corresponding to a non-RR r
Inputs e_r: variable matched with r in the RPG
$E_r = \{e_1, \ldots, e_l\}$: other endogenous variables in equation r
$X_r = \{x_1, \ldots, x_k\}$: exogenous variables in equation r
$C_r = \{c_1, \ldots, c_m\}$: component variables in equation r

1 $\hat{E}_r = \{\hat{e} \in \hat{E} : e \in E_r\}$
2 $DQR(r) = D(X_r) \times D(\hat{E}_r) \times D(C_r) \times D(\{e_r\})$
3 $\varphi_{nom} = (\forall \hat{e} \in \hat{E}_r : \hat{e}(nom)) \wedge (\forall x \in X_r : x(kwn)) \wedge (\forall c \in C_r : c(ok))$
4 $QR_{nom} = \sigma_{\varphi_{nom} \wedge e_r(nom)} DQR(r)$
5 $\varphi_{abk} = (\exists \hat{e} \in \hat{E}_r : \hat{e}(abk)) \wedge (\forall \hat{e} \in \hat{E}_r : \hat{e}(nom) \vee \hat{e}(abk)) \wedge (\forall c \in C_r : c(ok))$
6 $QR_{abk} = \sigma_{\varphi_{abk} \wedge e_r(abk)} DQR(r)$
7 $\varphi_{unk} = (\exists \hat{e} \in \hat{E}_r : \hat{e}(unk)) \vee (\exists c \in C_r : c(ab))$
8 $QR_{unk} = \sigma_{\varphi_{unk} \wedge e_r(unk)} DQR(r)$
9 $QR_r = QR_{nom} \cup QR_{abk} \cup QR_{unk}$

Fig. 2 Building Qualitative Relations for RRs.

- the set RES of residual variables; for each equation r in the RPG, there is a variable $res_r \in RES$ with possible values *abs* (absent), *z* (zero) and *nz* (non-zero). As we will see, a residual variable res_r takes the *abs* value only when the equation r is not a RR in the RPG and the endogenous variable e_r matched with r in the RPG is not sensorized

Note that we introduce a residual for each equation in the model since, by adding sensors, all of the equations may become RR and therefore have an associated residual. In the GFS system, the set of residuals RES, beside a residual res_{r5} for RR r_5, also contains residuals for the other equations $r_1 - r_4$ and $r_6 - r_{11}$; such residuals will take values different than *abs* when the hypothetical sensors of their matched variables are set to *yes*. For example, if hs_{p3} has value *yes* (i.e. p_3 is sensorized), relation r_1 becomes a RR and, therefore, the residual res_{r_1} will be allowed to take as value either *z* or *nz*, but not *abs*.

The next step consists in building a number of qualitative relations that specify the constraints among the C, X, E, RES, $\hat{E}$ and HS variables.

For each non-RR r, we build a qualitative relation QR_r which determines the value of the endogenous variable e_r matched with r in the RPG as a function of the values of the other (propagated) endogenous variables as well as the exogenous and component variables that appear in r (denoted respectively with $\hat{E}_r$, X_r and C_r). Figure 2 shows how QR_r is built. The need of expressing the value of e_r in terms of the values of the propagated endogenous variables $\hat{E}_r$ instead of the base endogenous variables E_r stems from the fact that each endogenous variable e that has an outgoing arc towards r in the RPG may have been observed with a sensor (see the description of hypothetical sensor relations below).

Relation QR_r contains tuples in which e_r has value *nom*, tuples where e_r has value *abk* and tuples where e_r has value *unk* (line 9).

The set QR_{nom} of tuples in which e_r has value *nom* is computed in lines 3-4 and contains all the assignments to $\hat{E}_r \cup X_r \cup C_r$ s.t. all the propagated endogenous

Algorithm **BuildHSR**
builds a qualitative hypothetical sensor relation HSR_r corresponding to a non-RR r
Inputs e_r: variable matched with r in the RPG
1 $DHS(r) = D(hs_{e_r}) \times D(e_r) \times D(\hat{e}_r) \times D(res_r)$
2 $\varphi_{abs} = hs_{e_r}(no) : HSR_{abs} = \sigma_{\varphi_{abs} \wedge (\hat{e}_r = e_r) \wedge res_r(abs)} DHS(r)$
3 $\varphi_z = hs_{e_r}(yes) \wedge (e_r(nom) \vee e_r(abk)) : HSR_z = \sigma_{\varphi_z \wedge (\hat{e}_r = e_r) \wedge res_r(z)} DHS(r)$
4 $\varphi_{nz} = hs_{e_r}(yes) \wedge e_r(unk) : HSR_{nz} = \sigma_{\varphi_{nz} \wedge \hat{e}_r(abk) \wedge res_r(nz)} DHS(r)$
5 $HSR_r = HSR_{abs} \cup HSR_z \cup HSR_{nz}$

Fig. 3 Building Hypothetical Sensor Relations for non-RRs.

variables have the nominal value *nom*, all the exogenous variables have the *kwn* value, and all the component variables are *ok* (note that there's only one assignment satisfying all these conditions, so QR_{nom} is a singleton).

Technically, the set QR_{nom} (as the other sets below) is computed by building a logic condition φ_{nom} equivalent to what we have informally stated above, and by selecting (with the σ operator of relational algebra) from the set $DQR(r)$ of all the possible tuples for r those tuples that satisfy φ_{nom} and $e_r(nom)$ (line 4).

Similarly, the set QR_{abk} of tuples in which e_r has value *abk* is computed in lines 5-6 and contains all the assignments to $X_r \cup \hat{E}_r \cup C_r$ s.t. all the component variables are *ok* and all the propagated endogenous variables have either the nominal value *nom* or an abnormal but known value *abk* but at least one of them has value *abk* (otherwise e_r would have a *nom* value). Indeed, provided all of the components associated with equation r are *ok*, if one or more propagated endogenous variables have abnormal (but known) values, the value of e_r that we predict by using r will be abnormal but it will be known.

Finally, the set of tuples QR_{unk} in which e_r has value *unk* is computed in lines 7-8 and contains all the assignments to $X_r \cup \hat{E}_r \cup C_r$ s.t. at least one component variable is *ab* or at least one propagated endogenous variable has an unknown abnormal value *unk*. Indeed, in such a case, the value of e_r that we predict by using r will be abnormal and unknown.

For the GFS system of Figure 1, we build a relation QR_r for all the equations $r_1, \ldots, r_{11}$, except for r_5 which is a RR.

For each non-RR r, we also build a qualitative hypothetical sensor relation HSR_r which expresses the relation between the endogenous variable e_r and its associated propagated variable $\hat{e}_r$ depending on the value of hypothetical sensor variable hs_{e_r}. Such a relation also determines the value of the residual res_r of r which is *abs* (absent) when the hypothetical sensor has value *no* (since, in such a case, r is a non-RR), but can take meaningful values when the hypothetical sensor has value *yes* (and r becomes a RR). Figure 3 shows how HSR_r is built.

Relation HSR_r contains tuples in which res_r has value *abs*, tuples where res_r has value z (zero) and tuples where res_r has value *nz* (non-zero) (line 5).

The set HSR_{abs} of tuples in which res_r has value *abs* (computed in line 2) contains the assignments where the hypothetical sensor hs_{e_r} has value *no*, as explained

above. In such a case, the value of the propagated variable $\hat{e}_r$ is set to be the same as the value of e_r (i.e. either *nom*, *abk* or *unk*).

The set HSR_z of tuples in which res_r has value z (computed in line 3) contains the assignments where hs_{e_r} has value *yes* and the endogenous variable e_r (which is sensorized) has either the nominal value *nom* or an abnormal but known value *abk*. Also in such a case, the value of the propagated variable $\hat{e}_r$ is set to be the same as the value of e_r (i.e. either *nom* or *abk*).

Finally, the set of tuples HSR_{nz} in which res_r has value nz (computed in line 4) contains only one assignment, where hs_{e_r} has value *yes*, the sensorized endogenous variable e_r has value *unk* and the propagated variable $\hat{e}_r$ has value *abk*. This is the case when, thanks to the presence of the sensor measuring e_r, the value of variable $\hat{e}_r$ (which is propagated to solve equations that are causally downstream in the RPG) becomes an abnormal but known value.

The third and last kind of relations we build are qualitative relations QRR_r which, for each RR r in the RPG, determine the value of the residual res_r as a function of the values of the exogenous, (propagated) endogenous and component variables that appear in r. Relations QRR_r are built in a similar way as relations QR_r and HSR_r, so we don't report the detailed algorithm because of lack of space.

For the example GFS system of Figure 1, we build a relation QRR_r only for equation r_5, which is the only RR in the RPG.

In summary, we have obtained a (qualitative) model starting from a numerical model of the system, where:
- *SV* (System Variables) is given by the sets C, X, E, HS, $\hat{E}$, RES, where $\hat{E}$ and RES can be considered just as special, additional endogenous variables
- *DT* (Domain Theory) is the composition (i.e. relational join) of relations QR_r, HSR_r and QRR_r defined above

The model we have built matches the notion of System Description SD introduced in section 2 for qualitative models, except for the presence of hypothetical sensors HS; we come back to the special role of HS variables in the next section.

While the relations QR_r, HSR_r and QRR_r have limited size since they are just local models involving a limited number of variables, the Domain Theory *DT* may be very large. For this reason it becomes of critical importance the ability of expressing *DT* in a compact form. In particular, following [7], in the present paper we have adopted Ordered Binary Decision Diagrams for encoding *DT* and other relations involved in our algorithms. For space and clarity reasons, we will keep expressing the algorithms in the following section in terms of relational algebra operations over extensional relations, instead of in terms of operations on OBDDs; the compilation of a relation with an OBDD and the mapping between relational algebra and OBDD operations is well known and quite straightforward (for an OBDD-based implementation of diagnosis see [8]).

5 Computation of MASS

The introduction of hypothetical sensors HS has made possible to parametrize the observability of a system, since now we are in the position of expressing any observability as an assignment of *yes* or *no* to the hypothetical sensors in the model. We denote with hs an assigment to HS variables, and with $O(hs)$ the set of endogenous variables e s.t. $hs_e(yes) \in HS$. The notion of discriminability can be reformulated in the following way.

Definition 4. Let hs be an assignment to HS variables and $c \in C$ be a system component. We say that c is *discriminable* w.r.t. observability $O(hs)$ iff for each instance x of X:

$$\Pi_{RES}(\sigma_{c(ok)\wedge x\wedge hs}(DT)) \cap \Pi_{RES}(\sigma_{c(ab)\wedge x\wedge hs}(DT)) = \emptyset$$

It is worth noting that, compared to Definition 1, the observability is determined by the assignment hs to the hypothetical sensor variables, and the comparison is made on the values of the residual variables RES. However, all the definitions based on indiscriminability, including those of MSS and MASS, are still valid.

Before describing in detail the computation of the MASS, we briefly summarize the main steps. The starting point is a set of discriminability requirements $\Delta = \{c_{\Delta,1}, \ldots, c_{\Delta,k}\}$ pointing out which components the user is interested to discriminate (for full diagnosability, Δ is equal to the whole set of system components).

Given a specific discriminability requirement $\delta_i = c_{\Delta,i}$, the system computes the set SS_{δ_i} which includes all the sensor sets that guarantee the discriminabilty of $c_{\Delta,i}$. We iterate the process for each discriminability requirement in Δ obtaining sets $SS_{\delta_1}, \ldots, SS_{\delta_k}$ and, by intersecting these sets, we get the set of all sensor sets SS_Δ which satisfy all the discriminability requirements in Δ.

At this point, the user can specify a set of constraints Ω on the sensors: more specifically, it is possible to specify that an endogenous variable e is certainly observed (by adding a constraint in Ω that assigns the value *yes* to hs_e), or that e is not to be considered (in this case the constraint in Ω assigns value *no* to hs_e). Note that Ω can be conveniently expressed as a partial assignment to the HS variables.

If the user puts no constraint in Ω, the system computes the MSS, otherwise it computes the MASS, according to definition 3.

Using SS_Δ and Ω, the minimization module is able to compute all the Minimum Additional Sensor Sets and therefore is able to provide the user with the MASS that satisfy his/her discrimination requirements.

It is worth noting that, in general, the globally optimal sets of sensors (i.e. the MASS for $\Delta = \{c_{\Delta,1}, \ldots, c_{\Delta,k}\}$ and Ω) can *not* be obtained as the union of the locally optimal MASS for requirements $c_{\Delta,1}$, …, $c_{\Delta,k}$, since such a union may be globally suboptimal. Therefore the optimization problem cannot be decomposed but has to be solved at the global level.

Computing SS_Δ. As stated above, we first compute SS_δ for each specific discriminability requirement $\delta = c$. This computation is summarized in Figure 4 (left) where the computational steps are expressed in terms of relational operations project

ComputeSSdelta(DT, c)
1 $H_{ok} = \sigma_{c(ok)}(DT)$
2 $H_{ok} = \Pi_{X \cup HS \cup RES}(H_{ok})$
3 $H_{ab} = \sigma_{c(ab)}(DT)$
4 $H_{ab} = \Pi_{X \cup HS \cup RES}(H_{ab})$
5 $H_{com} = H_{ok} \cap H_{ab}$
6 $SS_{com} = \Pi_{HS}(H_{com})$
7 $SS_{\delta} = D(HS) \backslash SS_{com}$

CompMASS(SS_{Δ}, Ω)
1 $SS_{\Delta,\Omega} = \sigma_{\Omega}(SS_{\Delta})$
2 i = 0
3 $MASS = SS_{\Delta,\Omega} \cap CSS_i$
4 **while** $MASS == \emptyset \wedge i < |E|$
5 i = i + 1
6 $MASS = SS_{\Delta,\Omega} \cap CSS_i$

Fig. 4 Computation of SS_{δ} (left) and of *MASS* (right).

and select. It is worth recalling that the actual implementation of the algorithm is based on OBDDs which encode the relations and operators working on such OBDDs.

Coming back to the algorithm, first of all we compute two relations H_{ok} and H_{ab} by restricting the domain theory DT to the cases when the component c is *ok* and to the complementary cases when the component c is *ab*.

Since we have to check that the two cases are discriminable in terms of observables, relations H_{ok} and H_{ab} are projected on variables which are relevant for discriminability, namely the variables X, RES, and HS.

Note that the behavioral modes of the other components are forgotten, since definition 1 requires that the *ok* and *ab* modes of c are discriminable regardless of the assignments of behavioral modes to the other components.

In line 5 we compute relation H_{com} by intersecting H_{ok} and H_{ab}. A tuple of relation H_{com} is an assignment $x \cup hs \cup res$ to the $X \cup HS \cup RES$ variables; the presence of such a tuple in H_{com} means that, when exogenous variables have value x and we observe *res* as the values of residuals, there exists at least one assignment to the component variables consistent with x and *res* where $c(ok)$ holds and at least another such assignment where $c(ab)$ holds.

In other words, a tuple $x \cup hs \cup res$ in H_{com} indicates that, at least under input x, $c(ok)$ and $c(ab)$ are not discriminable under the observability *hs*. Since Definition 1 requires that discriminability holds for all inputs x, we project H_{com} on HS variables in order to isolate the observabilities that violate this requirement (line 6).

By complementing this set, we obtain the set SS_{δ} of all the assignments to HS (i.e. possible sensor placements) that satisfy the requirement of discrimination for component c. This is done in line 7 by subtracting SS_{com} from $D(HS)$ which represents the set of all possible assignment to HS.

Once the sets SS_{δ_i} have been computed for each $\delta_i \in \Delta$, the set of sensors which satisfy all the discriminability requirements can be easily computed as $SS_{\Delta} = SS_{\delta_1} \cap \ldots \cap SS_{\delta_k}$.

Computing MASS from SS_{Δ}. The final step consists in computing the Minimum Additional Sensor Sets by exploiting SS_{Δ} and by taking into account the constraints provided by the user on the presence/absence of sensors (i.e Ω).

The computation of the MASS is performed by the function *CompMASS* (Figure 4 right). First of all (line 1) the algorithm takes into consideration the set of constraints Ω on the sensors provided by the user (which could be $\emptyset$).

The relation $SS_{\Delta,\Omega}$ contains now all the sensor sets satisfying Δ and compatible with the constraints in Ω. The minimization (lines 2-6) is performed by exploiting a set of precomputed sets of sensors CSS_i, where a generic CSS_i contains all the possible combinations of hypothetical sensors hs_e with exactly i sensors set to *yes*.

In other words each CSS_i represents all the possible sensor sets that involve the observation of exactly i variables. Therefore CSS_0 represents the case where nothing is observable (all the hs_e have value *no*) while $CSS_{|E|}$ represents the case when all the $|E|$ endogenous variables are actually observed (all the hs_e have value *yes*). Due to lack of space, we do not report the details of the computation of sets CSS_i; the algorithm for such a computation is reported in [7].

Given the sets CSS_i, the minimization step can be implemented in a very simple way (lines 2-6): it is sufficient to verify whether the intersection of $SS_{\Delta,\Omega}$ with CSS_i is not empty starting from CSS_0. As soon as we find a non-empty intersection for index i, relation *MASS* represents the set of all the possible combinations of i sensors which satisfy both the discriminability requirements and the constraints on the sensors. Note that i is guaranteed to be the minimum number of sensors since we have already verified that with $0, 1, \ldots, i-1$ sensors we fail in finding a solution.

Thanks to the fact that the OBDDs which encode sets CSS_i have a polynomial size in the number of endogenous variables E [8], it turns out that the potentially very expensive task of computing the MASS can be done in polynomial time w.r.t. the size of the OBDD $obdd(SS_\Delta)$ which encodes SS_Δ[3].

Property 1. Let $obdd(SS_\Delta)$ be an OBDD encoding set SS_Δ; then, $obdd(MASS)$ encoding all the MASS can be computed by *CompMASS* in time $O(|E|^3 \cdot |obdd(SS_\Delta|)$.

When the OBDD encoding SS_Δ is small, we have the guarantee that *CompMASS* can *always* find the MASS efficiently.

6 Application to the GFS

We have applied the approach described in the previous sections to the GFS system, whose RPG is reported in Figure 1. The experiments have been conducted with a Java implementation of the algorithms that uses the JBDD interface to the Buddy library for the OBDD operations; the test machine was equipped with an Intel Core Duo CPU at 2.4GHz and 2GB of RAM.

The OBDD representing the compiled *DT* has a size of 776 nodes, and is computed in less than 10msec. Although the example system is quite small (51 variables including the hypothetical sensor, residual and propagated endogenous variables),

[3] This property mirrors a similar result obtained for Minimum Cardinality Diagnoses whose proof is reported in [8].

776 nodes are a very limited size for representing all of the tuples of the (global) domain theory DT.

First, we compute the MASS under the same conditions of the example in [6]:
- we make the single fault assumption; in order to enforce this assumption, we intersect the OBDD which represents DT with an OBDD CD_1 representing all of the single fault diagnoses plus the assignment where all components are ok; OBDD CD_1 is computed in a similar way as the Cardinality Sensor Sets, the main difference being that the role of yes/no sensors is played by ok/ab components
- we let Ω be the set $\{hs_{fpg2}(yes), hs_{fqg}(yes), hs_{fsg}(yes), hs_{fsgr}(yes), hs_{q4}(no)\}$

The OBDD CD_1 has a size of 14 nodes, while the OBDD DT_1 representing $DT \cap CD_1$ has a size of 763 nodes, and is computed in less than 1msec.

We compute SS_δ for each requirement $\delta = c$, $c \in C$ and then intersect all of the SS_δ obtaining the set SS_Δ of all the sensor sets which satisfy all the requirements. The size of the OBDD representing SS_Δ is 63 nodes, while the maximum size of the sets SS_δ is 60 nodes. The time for computing SS_Δ starting from DT_1 is 20 msec.

Then, we intersect SS_Δ with Ω obtaining the OBDD for $SS_{\Delta,\Omega}$ (20 nodes) and compute the MASS from $SS_{\Delta,\Omega}$, obtaining one optimal solution $\{hs_{p3}(yes)\}$, which is the same as the one computed in [6]. These computations take less than 1msec.

As a variation, we also compute the MASS starting from the same SS_Δ, but with $\Omega = \emptyset$ (i.e. we compute the MSS under the single-fault assumption). It turns out that there is a unique MSS with cardinality 3, namely $\{hs_{fqg}(yes), hs_{q2}(yes), hs_{p3}(yes)\}$.

Another (more important) variation consists in releasing the single fault assumption and compute the sensor sets which guarantee diagnosability regardless of the number of faults affecting the system. In such a case, if Ω is set as in [6], we find the single MASS $\{q3(yes), p3(yes)\}$. This MASS, together with the 4 sensors required by Ω, results in the set of 6 sensors $\{hs_{fpg2}(yes), hs_{fqg}(yes), hs_{fsg}(yes), hs_{fsgr}(yes), hs_{p3}(yes), hs_{q3}(yes)\}$. The total time needed for this computation is around 20 msec.

As a last variation, we let $\Omega = \emptyset$ in the multiple-fault case, finding 3 different solutions of cardinality 6 among which a solution $\{hs_{fsg}(yes), hs_{q3}(yes), hs_{p3}(yes), hs_{fpg2}(yes), hs_{fqg}(yes), hs_{fsgr}(yes)\}$, which is the same sensor set as the one obtained above as the union of Ω and the MASS $\{q3(yes), p3(yes)\}$.

7 Conclusions

The problem of computing the MASS for numeric equations system models has been deeply investigated in the FDI literature (e.g. [6], [5], [3], [4]). In the present paper we have proposed and discussed a novel method for computing MASS by exploiting recent techniques based on the symbolic compilation of qualitative system models [7] within a structural approach suitable for numerical equations models.

Our work addresses the problem starting from a Resolution Process Graph which can be computed with some existing techniques developed within the FDI community (e.g. [6]); the RPG is then mapped to a qualitative relational model and symbolic AI techniques are applied in order to compute the MASS. In this respect, the paper

can be viewed as a bridge work across the AI and FDI approaches to model-based sensor placement; while other works have aimed at bridging these two approaches for related problems such as diagnosability [1], as far as we know the previous works do not directly address the computation of MASS.

The Domain Theory with hypothetical sensors defined in this paper has many similarities with the Extended HFS Matrix of [6], since they both record the relations between component failures and values of the residuals, conditioned by the set of sensors. However, *DT* is parsimoniously encoded as an OBDD and, thanks to this fact, it is able to define the failures-residuals relations not only under the single-fault assumption, but for the combinatorially larger space of multiple-fault situations.

The main advantage of the proposed approach consists in its flexibility that is made possible by the adoption of compilation techniques. In particular, we have shown that it is possible to compute and compactly represent a set SS_Δ which encodes all the sensor sets satisfying the given discriminability requirements. Once SS_Δ has been built, it is possible to perform a wide number of minimizations under different conditions with formal guarantees on the computational complexity.

The use of special variables for modeling the observability is not applicable just to numeric system models, but has been first developed for purely qualitative models. In particular, in previous papers (such as [7]) we have shown that the approach is applicable to qualitative systems of non-trivial size such as the *c74182* digital circuit from ISCAS85, involving 70 components and 28 hypothetical sensors.

While in the present paper we have used minimum cardinality of sensors as a preference criterion, an obvious generalization would be to allow different costs for the sensors. Our approach could be easily generalized to cover the case of a limited number of (qualitative) possible costs for each sensor by extending the notion of *CSS* relation of section 5.

References

1. Cordier, M.O., Travé-Massuyès, L., Pucel, X.: Comparing Diagnosability in Continuous and Discrete-Event Systems. In: Proc. Proc. DX, pp. 55-60, (2006)
2. Cassar, J.P., Staroswiecki, M.: A Structural Approach for the Design of Failure Detection and Identification Systems. In: Proc. IFAC Control of Industrial Systems (1997)
3. Travé-Massuyès, L., Escobet, T., Olive, X.: Diagnosability Analysis Based on Component-Supported Analytical Redundancy Relations. IEEE Tr. on Systems, Man and Cybernetics PART A **36**(6), 1146–1160 (2006)
4. Krysander, M., Frisk, E.: Sensor placement for fault diagnosis. IEEE Tr. on Systems, Man and Cybernetics PART A **38**(6), 1398–1410 (2008)
5. Commault, C., Dion, J., Agha, S.: Structural analysis for the sensor location problem in fault detection and isolation. In: Proc. IFAC World Congress, pp. 949-954 (2006)
6. Travé-Massuyès, L., Escobet, T., Milne, R.: Model-based Diagnosability and Sensor Placement. Application to a Frame 6 Gas Turbine Sub-System. In: Proc. IJCAI, pp. 551-556 (2001)
7. Torta, G., Torasso, P.: Computation of Minimal Sensor Sets from Precompiled Discriminability Relations. In: Proc. DX, pp. 202–209 (2007)
8. Torasso, P., Torta, G.: Model-Based Diagnosis through OBDD Compilation: a Complexity Analysis. LNCS **4155**, 287-305 (2006)

SHORT PAPERS

Artificial Immunity Based Cooperative Sustainment Framework for Multi-Agent Systems

R.C.M. Chan and H.Y.K. Lau[1]

Abstract Many studies show that the modelling concept of multi-agent systems (MAS) can be very useful for many industries, such as automated production systems, modern distribution centres and warehouses, port container terminals and transportation systems, etc. However, when applying them to real life where unpredictable factors exists that lead to agent failures, they will not be able to perform as expected or even failed completely. A MAS that can withstand and recover from unpredictable failures is much welcomed by many industries that adopt automation as an integral part of their businesses. Therefore, we propose a cooperative sustainment framework to help MAS to recover the failed agent nodes and extend the system life using artificial immunity inspired design. To verify the usefulness of the design, we carry out some experiments and the result is encouraging.

1 Introduction

In the past decade, much attention has been focused on multi-agent systems (MAS) and adaptive systems, which become increasingly important as a new paradigm for organizing Artificial Intelligent (AI) applications [1-5].

MAS typically consist of a group of autonomous agents in which each agent is coordinated to achieve a common goal for the group. As the number of working agents inside the system is one of the important factors to determine the cooperative achievements of MAS, any agent failures in the system would adversely affect the performance of MAS in terms of efficiency or effectiveness. That is why agent failures should be avoided.

Most of the previous empirical studies on MAS, however, had not involved possible factors leading to unpredictable agent failure. Those systems are highly

1 The University of Hong Kong
{chan.raymond,hyklau}@hku.hk

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_19,

vulnerable to any accident happened to agents in real life situation. Therefore, there is a need to develop effective mechanisms to protect the system from failure and help to restore the entire system functionality.

To address the mentioned problem, this paper presents a cooperative sustainment control framework which is inspired from biological theory of immunology to control the recovery process of multi-agent system with multiple agent nodes and optimize the trade-off between overall performance and maintainability. A simulation study is undertaken to verify the validity of the framework and the cooperative sustainment mechanism.

2 AIS-based Cooperative Sustainment Control Framework

Human immune system (HIS) is a complex, self-protected and self-maintained cooperative defence system in our body [6, 7] that ensures the survival of mankind for thousands of years by protecting our body from the invasion of foreign substances such as bacteria and virus. Behind this well-composed system inside our body, there are some sophisticated cooperative mechanisms that cooperate the trillions of immune cells to response to unpredictable invasion and attack of pathogens rapidly. By extracting the concepts behind HIS, we adopt as metaphors and engineering paradigms to solve different problems in real life. Systems using this immune-based metaphor are known as Artificial Immune Systems (AIS).

Inspired by the distributed and self-organized property of human immune systems [8, 9] and the earlier results [10-12], the Immune-Based Cooperative Sustainment Framework (IBCSF) presented in this paper is developed based on the immune network theory [13] and the reaction mechanism between innate and adaptive immune system. The core of the control framework is a two-layer behaviour control model with the concept of sustainment which exploits the immunity-based regulation mechanisms to control how the system restores the failed agent nodes and extends the system life while there exists some unpredictable random factors that lead to multiple agent failure.

2.1 Overview on the Control Architecture

HIS can be divided into innate defence and adaptive defence [14]. Innate immune defence serves as the first barrier which tries to stop pathogens entering our body and minimizes the chance of being infected; while the adaptive immune defence activates if pathogens evade from innate response and produce antibodies to fight and recover from infection [15]. Correspondingly, as depicted in Figure 1, IBCSF has two levels of sustainment response: 1) Self Sustain Response and 2) Cooperative Sustain Response, which provide indispensable defences for MAS to

recover from different degrees of failure and help to maintain the balance between system performance and sustainability.

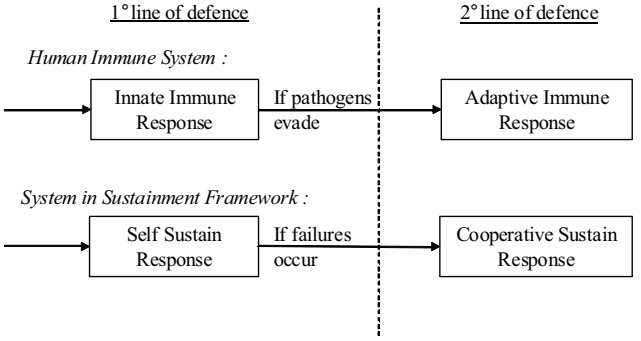


Figure 1 The two-level defence mechanism in HIS and the proposed sustainment framework

2.2 Sustainment Operations

Sustainment operations are defined as the obligatory behaviours that are used to protect and enhance the system sustainability, such that the system can function, maintain and remain productive for a longer period of time. Its definition is different from maintenance operations, as sustainment operations can include maintenance operations but not limited to repair and replacement. Rescheduling processes and reallocating resources can all be defined as a kind of sustainment operations as long as they help to extend the system life. A strategic sustainment operation series is known as sustainment response.

Self sustainment is the independent preservation strategic behaviour performed by individual agents; it is the first barrier of defence mechanism which stops and prevents an agent from failure. An example of self sustainment action can be a decision to stop partially damaged agent from its normal function and make it undergo repairing progress before it becomes completely malfunctioning. Self sustainment operations are considered as a less expensive repairing function, as the agents are still in a functional state and are able to trigger self sustainment response by themselves.

Cooperative Sustainment is the group preservation strategic behaviour performed by multiple agents, aiming to make the system survive in the environment with uncertain factors leading to serious unpredictable agent failures. This inter-preservation strategy starts whenever agent failure occurs in the system and helps repair the failed agents or take over the function of the failed agents. A typical example of cooperative sustainment action can be a rescue action, such as a functioning agent discovers a failed agent and sends it the repair station.

Cooperative sustainment operations are considered as the costliest function. As a failed agent is not able to trigger self-sustainment, the process requires the involvement of other functioning agents, in effect reducing the effective performance of the system at a particular period of time because the functioning agents are spending time and efforts to undergo maintenance instead of performing useful work.

3 Experiment and Result

To verify the sustainment ability of the proposed framework, we carried out several simulation studies with a 16 metres x 16 metres open space simulation platform using Player/stage [16] implemented under the Linux environment (see Figure 2). In the simulation platform, there are 50 randomly located tasks and a MAS formed by 10 mobile robot agents, each 0.4 metres wide by 0.5 metres long by 0.22 metres high. Each agent is installed with laser sensors that can detect items within a 1.2 metres radius.

There are background radiations that randomly damages the robots every process cycle and the power of the radiation is set to follow a normal distribution with a mean power of damaging 5%, 10%, 15%, 25% and 50% of a robot's health correspondingly in each experiment. The cooperative goal for the robots in the simulation environment is to discover and remove the tasks on the platform as much as possible while keeping most agents alive.

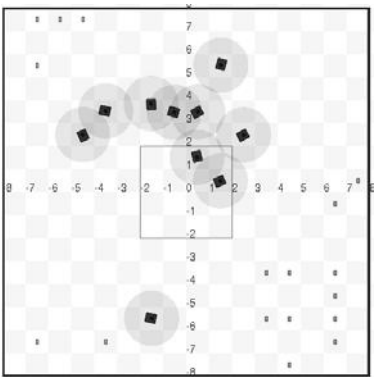


Figure 2 Ten robot agents searching for tasks in the simulation environment with stochastic damage rate inflicted by background radiation

The self sustainment operation is to bring the agent (robot) back to the 4 metres x 4 metres robot base area in the middle of the platform when the robot reach low health level, prevent it from fully damage. When a robot is brought back to the robot base, its health can be recovered 50% per cycle. Correspondingly, the

proposed cooperation sustainment operation is to send failed robot agents back to the robot base to restore their health.

The results of the experiment are shown in Table 1. The values above the brackets are the average number of working agents in the simulation environment for different damage robot health (measured as the corresponding percentage of damage health). The number inside the bracket represents the average system life (in terms of number of process cycle) before all agents failed. Setup A and Setup B are both control setups, where Setup A has no sustainment control implemented and Setup B has self sustainment control but no cooperative sustainment mechanism implemented. The simulation length is set to a maximum of 200 process cycle.

According to the results, the sustainment control mechanism has significantly extended the average system life in all tested environments. In Setup A, where no sustainment control is applied, only 53 process cycles are sustained by the robot agents in an environment with 15% mean damage rate. However, Setup C with full IBCSF design enables the system to survive for over 200 process cycles.

The control Setup B is used to compare the system sustainability if only the self sustainment control with first level of the sustainment mechanism is being applied. The results show the average number of working agents is relatively low, indicating that the average useful throughput of the system is rather limited. We can also see that using only one level of sustainment control is not sufficient to make the system to survive up to 200 process cycles. It is evidence that the incorporation of the second level is crucial for the overall sustainability.

Table 1. Simulation results under different percentages of robot health being damaged

	Environment with different mean Damage Ratio of robot health				
	5%	**10%**	**15%**	**25%**	**50%**
Setup A *No Sustainment Control*	5.335606 (53)	4.769078 (35)	4.580028 (24)	3.849418 (22.4)	4.12264 (14.6)
Setup B *IBCSF (With 1st level of Sustainment Control)*	3.818574 (134.8)	3.73316 (100.2)	3.225184 (100.4)	2.17701 (132.8)	3.953894 (32.8)
Setup C *IBCSF (With 2 level of Sustainment Controls)*	9.288059 (>200)	8.516416 (>200)	7.46766 (>200)	5.658979 (188.3)	3.913721 (76.6)

4 Conclusion and Future Works

In this paper, we introduce a new cooperative sustainment framework that is inspired from the human immune system and verified the validity of the proposed concepts using a multi-robot system performing cooperative tasks in a simulated environment. Although the study is in a rudimentary stage, the result of the

experiment is encouraging. The result shows the capability of framework to address the problem of agent failure modelled as a MAS.

Currently, we are investigating the impact of cooperative sustainment to the system overall performance. We are also enhancing the sustainment mechanism by incorporating a new agent communication scheme into the framework.

References

1. Alonso, E., D. Kudenko, and D. Kazakov (Eds.).: Adaptive Agents and Multi-Agent Systems — Adaptation and Multi-Agent Learning, Springer-Verlag, Berlin (2003).
2. Kudenko, D., D. Kazakov, and E. Alonso, (Eds.).: Adaptive Agents and Multi-Agent Systems II — Adaptation and Multi-Agent Learning, Springer-Verlag, Berlin (2005).
3. KES-AMSTA.: Agent and multi-agent systems — technologies and applications — first KES international symposium, KES-AMSTA 2007, Wroclaw, Poland, May 31-June 1, 2007 : proceedings, Springer-Verlag, Berlin (2007).
4. KES-AMSTA.: Agent and multi-agent systems: technologies and applications — second KES international symposium, KES-AMSTA 2008, Incheon, Korea, March 26-28, 2008, proceedingsn, Springer-Verlag, Berlin (2008).
5. KES-AMSTA.: Agent and multi-agent systems: technologies and applications — third KES international symposium, KES-AMSTA 2009, Uppsala, Sweden, June 3-5, 2009 ; proceedings KES-AMSTA 2009, Springer-Verlag, Berlin (2009).
6. Ishida, Y.: Immunity-Based Systems - A Design Perspective, Springer, Germany (2004)
7. Dasgupta, D.: Artificial Immune Systems and Their Applications, Springer, Berlin (1999)
8. Ishiguro, A., R. Watanabe, and Y. Uchikawa.: An immunological approach to dynamic behavior control for autonomous mobile robots. In: Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. IEEE/RSJ International Conference (1995)
9. Ko, A., H.Y.K. Lau, and T.L. Lau.: — General Suppression Control Framework: Application in Self-balancing Robots. in Artificial Immune Systems — 4th International Conference, ICARIS 2005, Banff, Alberta, Canada (2005)
10. Dasgupta, D.: An artificial immune system as a multi-agent decision support system. in Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on. (1998).
11. Lau, H.Y.K. and Wong, V.W.K.: An immunity-based distributed multiagent-control framework. Systems, Man and Cybernetics, Part A — Systems and Humans, IEEE Transactions on, Vol 36(1), pp. 91-108 (2006)
12. Lu, S. and H. Lau.: An Immunity Inspired Real-Time Cooperative Control Framework for Networked Multi-agent Systems. in Artificial Immune Systems: 8th International Conference, ICARIS 2009. York, UK (2009)
13. Jerne, N.K.: Towards a network theory of the immune system. Annales d'immunologie. 125C: pp. 373-389 (1974)
14. Male, D., et al. (Eds.).: Immunology. 7th International ed, Elsevier: Canada (2006)
15. Purves, W.K., et al.: Life — The Science of Biology. 6th ed, Sinauer Associates, Inc., USA (2001)
16. Player Project.: (2010) Available via http://playerstage.sourceforge.net/index.php?src=stage. Accessed 26 May 2010.

The Mining and Analysis Continuum of Explaining Uncovered

Martin Atzmueller and Thomas Roth-Berghofer

Abstract The result of data mining is a set of patterns or models. When presenting these, all or part of the result needs to be explained to the user in order to be understandable and for increasing the user acceptance of the patterns. In doing that, a variety of dimensions for explaining needs to be considered, e.g., from concrete to more abstract explanations. This paper discusses a continuum of explaining for data mining and analysis: It describes how data mining results can be analysed on continuous dimensions and levels.

1 Introduction

According to the CRISP-DM model [4] the data mining process consists of six phases: *Business Understanding* and *Data Understanding*, *Data Preparation*, *Modelling*, *Evaluation* and *Deployment*. These phases are ideally applied iteratively. In the evaluation phase the data mining models are checked and assessed by the user, before the models can be deployed: Often explanations for the complete models, or parts thereof are requested, e.g., for improving the acceptance of the patterns and their evaluation. Additionally, the mining process itself is a candidate for explanation, especially for inexperienced users. Appropriate explanation techniques in data mining and analysis are therefore crucial for an effective data mining approach; especially concerning *semantic data mining* and related approaches [2, 7], for which background knowledge provides further explanation capabilities.

Martin Atzmueller
University of Kassel, Knowledge and Data Engineering Group, Germany,
e-mail: atzmueller@cs.uni-kassel.de

Thomas Roth-Berghofer
German Research Center for Artificial Intelligence (DFKI) GmbH, Germany,
e-mail: thomas.roth-berghofer@dfki.de

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_20,

This paper presents the *mining and analysis continuum of explaining (MACE)*; see [3] for a detailled discussion. The starting point of explanation is given by the final and intermediate results of the data mining step. Also, the specification of the data mining task itself can often be iteratively refined guided by appropriate explanation of the results. This also provides for a consistent documentation of the process and design decisions involved, e.g., in the form of semantic analytical reports, cf. [2, 7]. The recipients of the explanation sessions are the data mining engineer and the end-user. For both appropriate explanations are provided depending on the user role: While the end-user is mainly concerned with the evaluation and deployment phases of the cycle, the data mining engineer is involved in the whole process.

The rest of the paper is structured as follows: Section 2 describes the basics of explanation-aware design and computing. Section 3 describes general explanation goals and kinds. After that, Section 4 outlines the MACE, including explanation-aware mining and analysis, and the continuum. Finally, Section 5 concludes the paper with a summary and discusses further interesting options for future research.

2 Explanation-Aware Software Design and Computing

Software systems need the ability to explain reasoning processes and their results as such abilities substantially affect usability and acceptance. Explanation-aware computing (ExaCt) is the vision of software systems being smart in interactions with their users. Explanation-aware Software Design (EASD) aims at guiding software designers and engineers to a purposeful explanation-aware software system by making their designers and engineers explanation-aware. The long-term goal is to provide the respective methods and tools for engineering and improving the explanation capabilities. Here we focus on bringing explanation-awareness to data mining.

Explanations are in some sense always answers to questions, may the questions be raised explicitly or not. They enhance the knowledge of communication partners in such a way that they understand each other better. Explanations support humans in their decision-making [11]. In a general explanation scenario we distinguish three main participants [10]: the *user* who is corresponding with the software system via its user interface, the *originator*, i.e., the problem solver or 'reasoning' component, which provides the functionality for the original task of the software, and the *explainer*. Originator and explainer need to be tightly coupled to help the explainer provide knowledge about the inner workings of the originator.

As introduced above, we distinguish certain user roles in the data mining context: the end-user and the data mining engineer. The end-user considers the process as the overall originator, i.e., the data mining system is the only originator. The data mining engineer also receives input from this originator, but we can also embed distinct originators into the individual steps of CRISP-DM. Then, each of those also contains an explanation component for the individual steps that can also contribute to the (global) originator for the end-user.

3 Goals and Kinds of Explanations

For application development, there are two immediately useful classifications of explanation: Goals and kinds. In designing a software system knowing about kinds of explanations helps with structuring available knowledge and deciding which knowledge further is required for exhibiting certain explanation capabilities. Spieker distinguishes several useful kinds of explanations for knowledge-based systems [13]. *Concept Explanations* answer such questions as 'What is *X*?' or 'What is the meaning of *X*?'. *Purpose explanations* describe the purpose of a fact or object. *Why Explanations* justify a fact or the occurrence of an event. *Action explanations* are a special case of why explanations. They explain or predict the behaviour of 'intelligent systems'. *How Explanations* are similar to action explanations. They describe the function of a device without an actual context.

Explanation goals help software designers focus on user needs and expectations towards explanations and help to understand what and when the system has to be able to explain (something). Sørmo et al. [12] suggest a set of explanation goals addressing transparency, justification, relevance, conceptualisation, and learning. In [9], Roth-Berghofer and Cassens outline the combination of both, goals and kinds of explanations, in the context of Case-Based Reasoning, and examine the contribution of the four CBR knowledge containers for modelling necessary knowledge. In the following we take up this idea and cast it on the field of data mining.

4 The Mining and Analysis Continuum of Explaining

The Mining and Analysis Continuum of Explaining (MACE) provides different perspectives on the same problem. It considers different goals and kinds of explaining, presentation modes, levels of detail of explanation, knowledge containers, and privacy. In the following, we first describe the data mining foundations of the MACE, before we discuss its explanation dimensions.

4.1 Explanation-Aware Mining and Analysis

We regard the data mining system as originator, and provide explanation capabilities for each of the phases of the CRISP-DM model. The involved mechanisms can be summarised as follows: The input of the system is given by a (descriptive) specification of the process, the (source) data, and optional background knowledge. The system output is given by a data mining model, e.g., a set of patterns. The output is then accompanied by a "description" of the elementary mining steps, i.e., traces and logs of the respective events and steps of the process. The output can then be explained in terms of input data, additional background knowledge and intermedi-

ate results (trace). Additionally, setting up the specification itself is often a difficult task, for which appropriate explanation features are crucial.

- In the *Business Understanding* phase (concept) explanation helps inexperienced users getting accustomed to the domain, by structuring the relations between the concepts, and explaining the concepts in terms of their properties. Especially ontological knowledge is thus helpful for explaining concepts and properties.
- In the *Data Understanding* phase, important data elements need to be selected. Then, missing or redundant attributes can be added or removed from the data set. This can be accomplished by a concept explanation step. Furthermore, known correlations/dependencies between concepts can then be uncovered.
- *Data Preparation* and *Modelling* are strongly connected: Both can benefit from concept and purpose explanations, for configuring/specifying the mining task, and preparing the data accordingly. Additionally, how explanations consider the mining process and can be used for justification and transparency of the process itself; they show how the results were actually derived.
- In the *Evaluation phase*, the discovered models/patterns need to be assessed by the user. Therefore, they need to be interpreted and explained in a structured way using the concepts and/or contained patterns. The discovered patterns, for example, can be matched to semantic relations or more complex relations between these. Additionally, such knowledge provides a potential (explaining) context for the discovered patterns. The results of the evaluation can then be utilised for *task refinement*, e.g., for adapting parameters and/or method settings.

4.2 Explanation Dimensions (Continuum)

As outlined above, we distinguish different dimensions of explanation (Figure 1). In the following, we discuss them briefly in the mining and analysis context.

The user and/or application *goals* relate mainly to the kind of explanation. During data mining, a data-driven approach starts with the (intermediate/final) results of the mining step. Then, explanation is provided by analysing the trace of the system. Transparency of the results can be significantly increased by using contextual, why, how, or purpose explanations.

The *presentation dimension* of explaining needs to be performed in an appropriate way, e.g., using textual information, aggregation such as tables or visualisations for more aggregation and abstraction. The design issues involved here are also strongly connected to the *detail dimension*, since the level of detail needs to be reflected by the presentation options and the presentation modes need to be compatible with the detail level. In the continuum, the presentation dimension provides seamless drill-down/roll-up capabilities similar to OLAP [6] techniques connected with the detail dimension.

The MACE makes use of different *knowledge containers*, cf., [5, 8] that include explicit knowledge for explaining. We distinguish the containers *ontological knowledge (vocabulary)*, *pattern knowledge*, *instance knowledge*, and *context knowledge*.

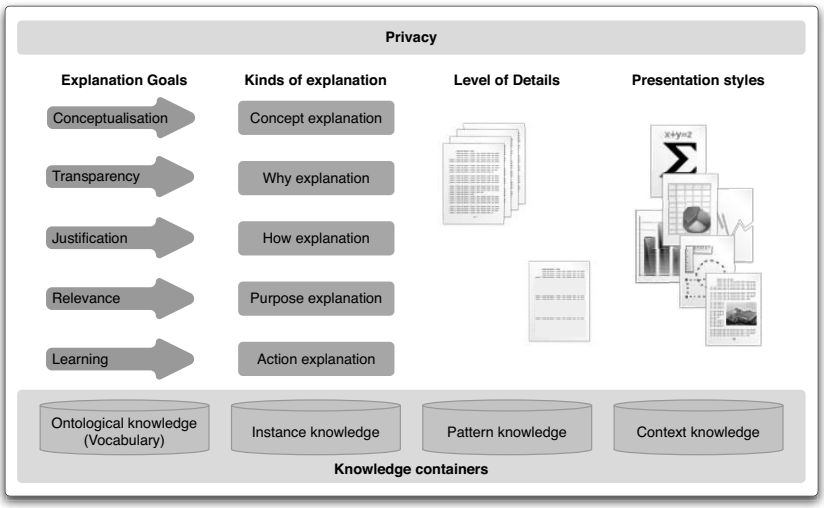


Fig. 1 Overview of the explanation dimensions

Whenever data is collected from heterogeneous sources, the aggregation of the data can reveal a lot more information than the single data sources. Privacy becomes an even more important issue with the availability and use of Linked (Open) Data.

In comparison to related application areas, e.g., case-based reasoning, the data mining and analysis domain provides for a more structured approach concerning the process, i.e., by applying the CRISP-DM cycle. In the individual steps of the process there are a variety of options for explanation, as discussed above. Additionally, the distinction between the 'inner' originators for the engineers and the 'outer' complete originator for the end-user, is also more present in the data mining context.

In practise, the proposed elements of the MACE need to be considered in a context of a specific data mining system. Additionally, the applied instantiation of the continuum also depends on the application domain. Both issues need to be considered when setting up the originator and explainer pair, and for arranging the match between them. Then, the utilisation of the instantiations of the dimensions depends significantly on the input context provided by the system, e.g., on the specification of the task, on the available trace information, and on the provided knowledge.

Since the knowledge containers are assigned both to the originator and the explainer, the specific knowledge containers can often be refined incrementally during the application of the system. While this is often easier considering the explainer, the extension and/or refinement of background knowledge applied by the data mining system is also possible. Several of the knowledge containers can often be reused 'as is' considering the originator, e.g., the ontological and instance knowledge containers. The pattern and context knowledge containers can usually be extended in the most flexible way, e.g., using Wiki-technology [1, 2].

5 Summary and Outlook

This paper presented a continuum of explaining for data mining and analysis: It described how data mining results can be analysed on several continuous dimensions and levels. We have described how the explanation options can be utilised in the standard *CRISP-DM* process model, and have briefly discussed the different goals and kinds of explanation in the context of the MACE.

For future work, we want to investigate ontological explanations in more detail, especially in the context of ubiquitous and social environments. Furthermore, appropriate tool support is necessary, especially regarding the presentation dimensions. Therefore, we want to investigate advanced explanation-aware presentation techniques in the context of the KNOWTA [1, 2] system, focusing on the concrete explanation-enhancing design issues.

References

1. Atzmueller, M., Haupt, F., Beer, S., Puppe, F.: Knowta: Wiki-Enabled Social Tagging for Collaborative Knowledge and Experience Management. In: Proc. Intl. Workshop on Design, Evaluation and Refinement of Intelligent Systems (DERIS), vol. CEUR-WS 545 (2009)
2. Atzmueller, M., Lemmerich, F., Reutelshoefer, J., Puppe, F.: Wiki-Enabled Semantic Data Mining - Task Design, Evaluation and Refinement. In: Proc. Intl. Workshop on Design, Evaluation and Refinement of Intelligent Systems (DERIS), vol. CEUR-WS 545 (2009)
3. Atzmüller, M., Roth-Berghofer, T.: Ready for the MACE? The Mining and Analysis Continuum of Explaining uncovered. Research Report RR-10-02, Deutsches Forschungszentrum für Künstliche Intelligenz (2010). ISSN 0946-008X
4. Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R.: CRISP-DM 1.0: Step-by-Step Data Mining Guide. CRISP-DM consortium: NCR Systems Engineering Copenhagen (USA and Denmark) DaimlerChrysler AG (Germany), SPSS Inc. (USA) and OHRA Verzekeringen en Bank Groep B.V (The Netherlands) (2000)
5. Clancey, W.J.: The Epistemology of a Rule-Based Expert System: A Framework for Explanation. Artificial Intelligence **20**, 215–251 (1983)
6. Han, J., Kamber, M.: Data Mining: Concepts and Techniques, 2nd Edition. Morgan Kaufmann, San Francisco, USA (2006)
7. Kliegr, T., Ralbovsky, M., Svatek, V., Simunuk, M., Jurkovsky, V., Nerava, J., Zemanek, J.: Semantic Analytical Reports: A Framework for Post-processing Data Mining Results. In: ISMIS 2009: Foundations of Intelligent Systems, no. 5722 in LNAI, pp. 88–98. Berlin (2009)
8. Richter, M.M.: The knowledge contained in similarity measures. Invited Talk at the First International Conference on Case-Based Reasoning, ICCBR'95, Sesimbra, Portugal (1995)
9. Roth-Berghofer, T.R., Cassens, J.: Mapping goals and kinds of explanations to the knowledge containers of case-based reasoning systems. In: Case-Based Reasoning Research and Development, Proc. 6th Intl. Conf. on Case-Based Reasoning, no. 3620 in Lecture Notes in Artificial Intelligence LNAI, pp. 451–464. Springer Verlag, Heidelberg (2005)
10. Roth-Berghofer, T.R., Richter, M.M.: On explanation. Künstl. Intelligenz **22**(2), 5–7 (2008)
11. Schank, R.C.: Explanation Patterns: Understanding Mechanically and Creatively. Lawrence Erlbaum Associates, Hillsdale, NJ (1986)
12. Sørmo, F., Cassens, J., Aamodt, A.: Explanation in case-based reasoning – perspectives and goals. Artificial Intelligence Review **24**(2), 109–143 (2005)
13. Spieker, P.: Natürlichsprachliche Erklärungen in technischen Expertensystemen. Dissertation, University of Kaiserslautern (1991)

Genetic Folding: A New Class of Evolutionary Algorithms

M.A. Mezher[1] and M.F. Abbod[2]

Abstract In this paper, a new class of Evolutionary Algorithm (EA) named as Genetic Folding (GF) is introduced. GF is based on novel chromosomes organisation which is structured in a parent form. In this paper, the model selection problem of Support Vector Machine (SVM) kernel expression has been utilised as a case study. Five UCI datasets have been tested and experimental results are compared with other methods. As a conclusion, the proposed algorithm is very promising and it can be applied to solve further complicated domains and problems.

1 Introduction

Support Vector Machine (SVM) algorithm has been implemented effectively in many application domains, such as classification and regression [3]. However, the model selection problem has not been successfully addressed for practical application,. Previous works have been mostly dedicated to either adapting the kernel functions or tuning its models [14]. For this purpose, different approaches have been presented which combine standard SVM with the Genetic Algorithms (GA) [10, 11, 12] or with Genetic Programming (GP) [4, 5, 6, 7, 9]. Such approaches have produced new kernel functions either subjected to the Mercer's rules [4, 5, 6, 10] or not [7].

Genetic Folding (GF) is an inherited class of an Evolutionary Algorithm (EA). GF like GA and GP is based on populations of individuals [1, 2]. However, the mechanism of RNA string is simulated, which folds every base pair with its complementary [15]. In this paper, GF is presented to optimize new Mercer's rule satisfying kernel function designed for binary classification problems.

[1] Brunel University, West London, Uxbridge, UB8 3PH, UK. mohd.mezher@brunel.ac.uk
[2] Brunel University, West London, Uxbridge, UB8 3PH, UK. maysam.abbod@brunel.ac.uk

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_21,

2 Problem Definition

SVM classifies data by determining a set of support vectors, which are members of the training inputs that outline a hyperplane in feature space [4]. The classifier has the properties of maximizing the margin and minimizing the generalization error, which is based on the chosen kernel. In order to improve the classifier's generalization capability, the kernel function is building a linear machine instead of a nonlinear machine with the same computational cost. For a problem in binary classification suppose $T = (x_i, y_i)$, where $x_i \in \Re^n$ represents the input data and each $y_i, y_i \in \{-1,+1\}$ is an output (label). The classifier can be evaluated efficiently using the inner product of the test $\phi(x)$ and training $\phi_i(x)$ points as follows:

$$F(x) = \sum_{i=1}^{N} \alpha y K < \phi_i(x), \phi(x) > -b + \xi \geq 1 = 1 \tag{1}$$

where α represents the Lagrangian vector, b is the bias value, ξ is the slack variable and *K* is any type of kernel functions. Defining a kernel function is a trivial task compared to creating a complicated feature of nonlinear space. However, the kernel functions must have properties that are necessary to be satisfied for some feature space; such as symmetry; i.e. $K(x,z) = K(z,x)$, and Mercer's theorem satisfaction; i.e. $K = (K(x_i, x_j))_{i,j=1}^{n}$ being a positive semi-definite [3]. For more details about kernel function problems, the reader may refer to [3, 13]. In this paper, GF is utilised to generate a new kernel matrix that satisfy these rules. In order to make the GF kernel symmetrical, dot products of the GF kernel is used with the reverse order of the same GF kernel.

3 Genetic Folding Algorithm

GF life cycle is similar to EA which starts with generating a number of random individuals. The valid chromosome (arithmetic operation) will be encoded to be represented in the initial population. Next, GF decoded all valid individuals to be evaluated. Afterwards, the chromosome are then selected by the roulette wheel depending on the given fitness value. After that, the fittest chromosomes are selected; subject to the genetic operators; to generate new populations in an independent way. The genetic operators used in this works are single-point crossover and swap mutation operator. The whole process is repeated each generation until the optimum chromosome (kernel) is achieved.

3.1 Genetic Folding Programming

The main component of the GF chromosome is the genes, which comprises of three parts; the gene's index (father) and two parts inside the gene (children). Figure 1(a) is an example of kernel function. The GF gene's structure has two components to be considered; the left side (ls) part and the right side (rs) part which represent the left and the right child respectively.

1	2	3	4	5	6	7	8	9	0	
+	-	*	p_1	/	x_i	x_j	p_2	cos	y	(a)
2.3	4.5	8.9	p_1	6.7	x_i	x_j	p_2	10	y	(b)

Figure 1. An example of GF chromosome

However, there are three types of fathers' relationship; two children (two operands), one child (one operand) and no child (terminal). In this case, GF represents two operands, one operand, no operands as a float, constant, and zero number respectively. Therefore, each father takes his children according to his type. Consequently, the operators are arranged in an ascending order from the highest to the lowest term. Obviously, in figure 1(a), the plus operator takes the first place and the y terminal takes the last place.

3.2 Encoding and Decoding Procedures

Figure 2 shows the whole process of encoding/decoding steps of the problem in Figure 1(a). The GF algorithm maps the genes' indices (operator) to its complementary operands. The gene's index represents the father's number in the chromosome. The offspring comprises two digits, the ls and the rs which are separated by dots in the middle. Obviously, Figure 2 forms the kernel function introduced in Figure 1(a). Note that, the float numbers inside the gene referred to the folded children over the father (index).

To encode an arithmetic operation, the process starts at index 1 (plus operator) which has two children (2, 3); and the ls and rs children. The ls child is the value in index 2. Consequently, the ls child (number 2) becomes a father with two children (4, 5). On the other side, the rs child of the plus operator is the value in index 3 which has two children (8, 9). Repeatedly, the GF algorithm will carry on representing the rest of the operators and operands until a zero value (terminal) is reached in both sides.

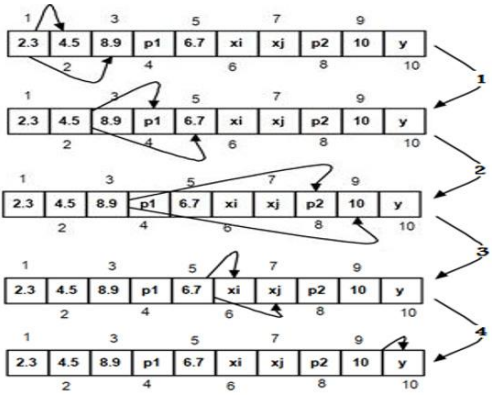


Figure 2. Encode/Decode Genetic Folding

In the meantime, to decode the chromosome, the first gene which has two divisions is considered. The first father is the plus operator which has two operands (2.3); the ls child (minus) and the rs child (multiply). This process is defined as folding, due to the way which folding of the ls child (minus) and the rs child (multiply) over the father cell (plus) occurs. Repeatedly, for each father, there are a number of children to be called every time.

4 Experimental Design and Results

GF algorithm is applied to five binary classification datasets. The two fitness functions used are the correctness rate and the Area Under the Curve (AUC) [8]. The data is divided into 75% training and 25% testing [3]. Each experiments is repeated for five times for testing reliability and efficiency of the proposed system. The UCI datasets (http://archive.ics.uci.edu/ml) used are Ionosphere (351×34), Pima Indians (768×8), Bladder Cancer (693×12), Cleve (297×13) and BUPA Liver (345×6).

For comparison purposes, different architecture were tested, standard SVM, GA tuned SVM, GP based SVM and the proposed architecture. For the standard SVM, random values have been initialized. However, SVM-GA applies a binary representation to tune the two parameters in the polynomial function. The SVM-GP was developed using GPLAB toolbox [14].

However, in proposed GF architecture, five arithmetic operators were included. The arithmetic operators are (sine, cosine, plus, minus, multiply). The GF gene comprises of two digits in each side (ls. rs) which resulted in 198 bit chromosome length. Two types of genetic operators were used; swap mutation, single point recombination genetic operators [1]. The single-point crossover probability was set to $p_c \geq 0.6$ and the swap mutation operator was $p_m \leq 0.05$.

Tables 1 and 2 show the fitness value that have been drawn for all the dataset. The fitness function has been computed for both AUC (Table 1) and correctness rate (Table 2). The experimental results were taken with the regularization parameter C=3. An example of the fittest kernel function founded for the Bupa dataset is shown in Table 3.

Method / Data	SVM	GA	GP (without Mercer's rules)	GF (with Mercer's rules)
BUPA Liver	51.09± 1.7	57.07± 1.8	57.89± 0.81	64.55± 1.1
Ionosphere	62.82± 4.7	67.45± 1.3	73.65± 1.3	89.87± 1.2
Bladder Cancer	53.58± 1.2	59.13± 1.4	57.19± 1.3	69.27± 1.1
Cleveland	66.95± 1.3	78.93± 2.3	78.31± 1.4	81.94± 2.1
Pima Indians	62.25± 0.2	62.48± 2.1	64.27± 1.6	70.55± 1.8

Table 1. AUC ± standard deviation for five UCI datasets

Method / Data	SVM	GA	GP (without Mercer's rules)	GF (with Mercer's rules)
BUPA Liver	58.45 + 1.7	61.45 + 1.8	65.11 + 0.81	76.73 + 1.1
Ionosphere	58.86 + 4.7	81.71 + 1.3	87.62 +1.3	95.23 +1.2
Bladder Cancer	68.50+1.2	68.50+1.4	76.11+1.3	76.31+1.1
Cleveland	72.97+1.3	85.81+2.3	85.15+1.4	87.81+2.1
Pima Indians	45.19 +4.2	69.79 +1.6	75.63 +2.1	79.15 + 1.8

Table 2. Correctness rate ± standard deviation for five UCI datasets

The Equation	((X2+X)+X2) × ((X+X2)+X)
Genetic Folding	(2.7)(3.6)(4.5)(0.0)(0.0) (0.0)(8.11)(9.10)(0.0)(0.0) (0.0)

Table 3. GF Chromosome of Bupa Dataset

However, the efficiency of the classifier depends on various issues that affect the classifiers' results. Such issues are; the number of the operators, free parameters, the cross-validation methods, the population and chromosomes size and the kernel function.

5 Conclusions

Genetic Folding is a novel algorithm inspired by the folding mechanism in the RNA sequence. The proposed GF algorithm has produced an accurate classifier for SVM binary classification problems.

The advantages of GF come from its nature and can be stated in five points: First, the chromosomes are straightforward entities and simple structure. Second, GF has a diverse length of chromosomes to be transmitted to the next generation; therefore, GF genetic operators do not depend on the numbers of the genes in each chromosome. Third, the gene has different types of children from each father (index). Forth, GF can be developed for model selection without the need for extra computation. Fifth, GF can be applied to other problems such as multi-classification and regression.

References

1. Sivanandam, S. and Deepa, S., 'Introduction to Genetic Algorithm', Springer, 15-130 (2008).
2. Koza, J. R., 'Genetic Programming: on the Programming of Computers by Means of Natural Selection', 74-147, Cambridge, MA: The MIT Press, (1992).
3. Cristianini, N. and Shawe-Taylor, J., 'An Introduction to Support Vector Machines: and Other Kernel-Based Learning Methods', 1st ed. Cambridge University Press, (2000).
4. Dioşan, L., Rogozan, A. and Pecuchet, J-P., 'Optimising Multiple Kernels for SVM by Genetic Programming', Evolutionary Computation in Combinatorial Optimization, vol. 4972, 230-241 (2008).
5. Sullivan, K. and Luke, S., 'Evolving Kernels for Support Vector Machine Classification', Genetic And Evolutionary Computation Conference, 1702 – 1707 (2007).
6. Gagné, C., Schoenauer, M., Sebag, M. and Tomassini, M., 'Genetic Programming for Kernel-based Learning with Co-evolving Subsets Selection', LNCS, no. 4193, 1008-1017 (2006).
7. Howley, T. and Madden M., 'The Genetic Kernel Support Vector Machine: Description and Evaluation', Artificial Intelligence Review, vol. 24, no. 3-4, 379-395 (2005).
8. Fawcett, T., 'An Introduction to ROC Analysis', Pattern Recognition Letters, vol. 27, no. 8, 861-874 (2006).
9. Diosan, L., Rogozan, A. and Pecuchet, J-P. , 'Evolving Kernel Functions for SVMs by Genetic Programming', Machine Learning and Applications, ICMLA, 19-24 (2007).
10. Chen, P-W., Wang, J-Y. and Lee, H-M., 'Model Selection of SVMs Using GA Approach', IEEE International Joint Conference, vol. 3, 2035- 2040 (2004).
11. Staelin C., 'Parameter Selection for Support Vector Machines', HP Laboratories, (2003).
12. Lessmann, S., Stahlbock, R. and Crone, S. F., 'Genetic Algorithms for Support Vector Machine Model Selection', Proc. of the Intern. Joint Conf. on Neural Networks (IJCNN'06), Vancouver, Canada, (2006).
13. Rojas, S.A. and Fernandez-Reyes, D., 'Adapting Multiple Kernel Parameters for Support Vector Machines using Genetic Algorithms', IEEE, vol. 1. 626-631 (2005).
14. Silva S., 'GPLAB: A Genetic Programming Toolbox for MATLAB', (2007).
15. "Genes & Gene Expression". The Virtual Library of Biochemistry and Cell Biology. BioChemWeb.org. 2010-02-08.

SOMA: A Proposed Framework for Trend Mining in Large UK Diabetic Retinopathy Temporal Databases

Vassiliki Somaraki[1], Simon Harding[2], Deborah Broadbent[2], Frans Coenen[3]

Abstract In this paper, we present SOMA, a new trend mining framework; and Aretaeus, the associated trend mining algorithm. The proposed framework is able to detect different kinds of trends within longitudinal datasets. The prototype trends are defined mathematically so that they can be mapped onto the temporal patterns. Trends are defined and generated in terms of the frequency of occurrence of pattern changes over time. To evaluate the proposed framework the process was applied to a large collection of medical records, forming part of the diabetic retinopathy screening programme at the Royal Liverpool University Hospital.

1 Introduction

Trend mining is the process of discovering interesting trends in large time stamped datasets. The approach to trend mining advocated in this paper is to measure changes in frequently patterns that occur across time stamped (longitudinal) datasets. The focus of this paper is the longitudinal diabetic retinopathy screening data collected by the Royal Liverpool University Hospital (RLUH), a major centre for retinopathy research. The challenges of this particular data set are: (i) that it is large and complex, 150,000 episodes, comprising some 450 fields (of various types); (ii) it does not fit into any standard categorisation of longitudinal data in that the "time stamp" used is the sequential patient consultation event number

[1] Department of Computer Science, University of Liverpool, UK, L69 3BX and, University of Liverpool, L69 3GA, UK. V.Somaraki@liverpool.ac.uk

[2] Ophthalmology Research Unit, School of Clinical Science, University of Liverpool, L69 3GA, UK, and St. Paul's Eye Unit, Royal Liverpool University Hospital, L7 8XP, UK. {sharding,D.M.Broadbent}@liverpool.ac.uk

[3] Department of Computer Science, University of Liverpool, UK, L69 3BX. Coenen@liverpool.ac.uk

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_22,

where the duration between consultations is variable; and (iii) the data, in common with other patient datasets, contains many empty fields and anomalies. This last issue was addressed by developing a set of logic rules. In the context of empty fields the logic rules were used to define where values were not relevant and where data was incomplete. In the case of inter-related data, the logic rules were used to derive additional fields providing relevant definitions. To identify trends in the form of longitudinal data a trend mining framework was developed, SOMA, together with an associated trend mining algorithm (Aretaeus). Both are described in this paper.

2 Diabetic Retinopathy Databases

Diabetic Retinopathy (DR) is the most common cause of blindness in working age people in the UK. DR is a chronic multifactorial disease affecting patients with Diabetes Mellitus and causes damage to the retina [4]. Over 3,000,000 people suffer from diabetes and at least 750,000 of these people are registered blind or partially sighted in the UK. The remainder are under the risk of blindness. The RLUH has been a major centre for retinopathy research since 1991. Data collected from the diabetic retinopathy screening process is stored in a number of databases. The structure of these databases, and the tables that comprise them, reflect the mechanism whereby patients are processed, and also includes historical changes in the process [1]. The Liverpool Diabetic Eye Screening Service currently deals with some 17,000 people with diabetes registered, with family doctors, within the Liverpool Primary Care Trust per year. Consequently, a substantial amount of data is available for analysis.

3 The SOMA Trend Mining Framework

Figure 1 depicts the operation of the SOMA framework from the input of data, via the Aretaeus algorithm, to the final output. The raw data first goes to the warehouse; and then to the Data Pre-processing Software where data cleansing, creation of data timestamps, selection of subsets for analysis and the application of logic rules takes place. The data, after pre-processing, then goes to the data normalization stage, after which the frequent patterns are generated by applying the Total From Partial (TFP) frequent pattern mining algorithm [2,3] to every episode (defined by a unique time stamp) in the given data set. Then the frequent patterns and their frequency of occurrence are passed to Aretaeus algorithm to apply trend mining in order to produce different kind of prototype trends across the datasets based on the changes of the support.

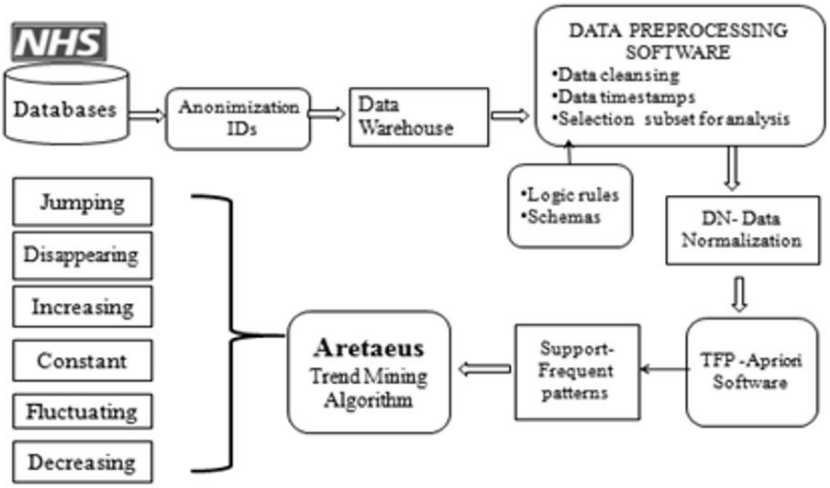


Figure 1: Representation of SOMA Framework

The Aretaeus algorithm uses mathematical identities (prototypes) to categorize trends. Let I be a frequent item set, identified within a sequence of time stamped data sets D_1, D_2… D_n, with support values of S_1, S_2, ..., S_n (where n is the number of timestamps). The *growth rate* (GR) associated with a trend is then defined as:

$$GR = \sum_{i=1}^{n-1} \frac{S_{i+1} - S_i}{S_i} + 1$$

The mathematical identities used by Aretaeus are presented in Table 1. The Aretaeus algorithm comprises the following basic steps:

1. Read, as input, the frequent patterns and their support values generated by the TFP algorithm.
2. Define the trends as vectors where the length of each vector is equal to the number of time stamps, so each element of the vector represents a time stamp.
3. Where the support for an itemset, at any time stamp is less than the support threshold, the support value is recorded as 0.
4. Categorize the trends according to a predefined set of trend prototypes (see Table 1) to create clusters (groups) of trends.

With reference to Table 1 the Jumping and Disappearing trends can be categorized further by considering trend sub-sequences. For example a Jumping trend can be Jumping-Increasing, Jumping-Constant or Jumping-Decreasing.

Similarly the increasing, constant and decreasing categories can be combined by pairing trend sub-sequences as shown in Table 2.

Table 1. Trend Categorisation Identities

Type	Mathematical conditions
Increasing (Inc)	$\frac{S_{i+1}}{S_i} > 1, \forall\, i \in [1, n-1]$, GR>ρ
Decreasing (Dec)	$\frac{S_{i+1}}{S_i} < 1, \forall\, i \in [1, n-1]$
Constant (Const)	$\frac{S_{i+1}}{S_i} = 1 \pm k, \forall\, i \in [1, n-1]$, k : tolerance threshold
Fluctuating (Fluct)	$\frac{S_{i+1}}{S_i} = 1 \pm k, \forall i \in [1,n-1]$ *and* $\frac{S_{j+1}}{S_j} > 1, \forall i \in [1,n-1], j \neq i$ $\frac{S_{i+1}}{S_i} = 1 \pm k, \forall i \in [1,n-1]$ *and* $\frac{S_{j+1}}{S_j} < 1, \forall i \in [1,n-1], j \neq i$ $\frac{S_{i+1}}{S_i} > 1, \forall i \in [1,n-1]$ *and* $\frac{S_{j+1}}{S_j} < 1, \forall i \in [1,n-1], j \neq i$ $\frac{S_{i+1}}{S_i} > 1, \forall i \in [1,n-1]$ *and* $\frac{S_{j+1}}{S_j} < 1, \forall i \in [1,n-1], j \neq i$ *and* $\frac{S_{l+1}}{S_l} = 1 \pm k, \forall l \in [1,n-1]\ l \neq j, l \neq i$
Jumping (Jump)	$for\ m < n: \quad S_i = 0, \forall\, i \in [1, m] \quad and \quad S_i > 0\ \forall\, i \in [m+1, n]$
Disappearing (Disp)	$for\ m < n: \quad S_i > 0, \forall\, i \in [1, m] \quad and \quad S_i = 0\ \forall\, i \in [m+1, n]$

Table 2. Combinations of Increasing, Decreasing and Constant of trends subsequences

	Increasing	Decreasing	Constant
Increasing	Inc	Inc-Dec	Inc-Const
Decreasing	Dec-Inc	Dec	Dec-Const
Constant	Const-Inc	Const- Dec	Const

4 Experimental Evaluation

This section presents an evaluation of SOMA. The evaluation was directed at an analysis of: (i) the number of trends that might be discovered and (ii) the nature of the trend categorisation. The RLUH Diabetic Retinopathy database was used for the evaluation. The RLUH database has recorded details of some 20,000 patients spanning an eighteen year period. Patients with diabetes are screened annually. Patients enter and leave the screening programme at different times. The average time that a patient spends within the screening process is currently six years. Thus, for the evaluation, only those patients that had taken part in the programme for at least six years were selected. Where patients had been in the programme for more than six years, data from the first six consultations was selected. This gave a dataset comprising six time stamps with 1430 records per time stamp. 7 data fields

were used for the evaluation, which, after normalisation and discretisation, resulted in 215 attributes. It is worth noting that the data required significant "cleansing" to remove noise and to address the issue of empty fields.

Table 3 presents an analysis, using a sequence of support thresholds (S), of: (i) the total number of trends generated, (ii) the number of trends in each category and (iii) the run time in seconds required by the trend mining software to generate and categorise the trends. The k tolerance threshold was set to 0.05, and the ρ growth/shrink rate threshold to 1.1. It is interesting to note that no constant trends were identified (because the nature of the K threshold value used). Figures 2 to 5 plot the data presented in Table 3 so as to demonstrate the increase in the number of trends, assigned to the six categories (prototypes), as the value for S is reduced. Inspection of the figures indicates, as expected, that the number of trends decreases as the support threshold increase. Note that in Figures 2 to 5 the X-axis represents a sequence of support thresholds and the Y-axis the number of Increasing, Decreasing, Total, and Fluctuating trends respectively.

Table 3. Trend Mining Framework Evaluation (p = 1.1, k = 0.05)

Support T'hold	Number of Trends						Total Num. Trend	Run Time (sec)
	Inc	Dec	Const	Disp	Fluct	Jump		
0.5	14	25	0	1827	930	1376	7602	2928.62
1.0	12	12	0	714	638	559	2532	1154.25
2.5	1	2	0	235	134	193	874	410.93
5.0	0	3	0	74	11	59	266	188.99
10.0	0	6	0	25	3	25	108	69.08

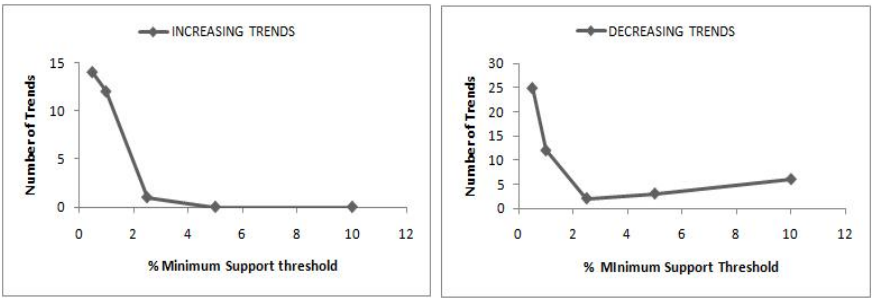


Figure 2, 3: Number of Increasing and Decreasing Trends vs. Minimum Support Threshold

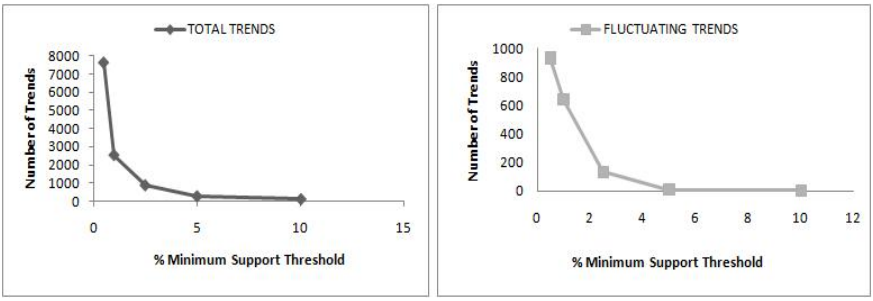


Figure 4, 5: Number of Total and Fluctuating Trends vs. Minimum Support Threshold

5 Conclusion

In this paper, we have described a novel approach to mine trends from a large amount of data. The Aretaeus algorithm allows generating more than 20 different kinds of trends and is able to discover hidden, useful information across the datasets. The fundamental idea underlying this paper is to use the support values of item sets across datasets in order to indentify useful trends. The advantage of this method is the classification of trends into categories, which is ideal for large databases. Finally, the development of a mechanism for the appropriate representation of the results using Bayesian networks is a topic of ongoing work, which will be particularly suitable for this purpose.

References

1. Somaraki, V., Broadbent, D., Coenen, F. and Harding, S.: Finding Temporal Patterns in Noisy Longitudinal Data: A Study in Diabetic Retinopathy. Proc. 10th Ind. Conf. on Data Mining, Springer LNAI 6171, pp418-431 (2010).
2. Coenen, F.P., Leng, P. and Ahmed, S.: Data Structures for association Rule Mining: T-trees and P-trees. IEEE Transactions on Data and Knowledge Engineering, Vol 16, No 6, pp774-778 (2004).
3. Coenen, F.P. Leng, P., and Goulbourne, G.: Tree Structures for Mining Association Rules. Journal of Data Mining and Knowledge Discovery, Vol 8, No 1, pp25-51 (2004).
4. Kanski, J.: Clinical Ophthalmology: A systematic Approach. Butterworth-Heinemann/Elsevier (2007).

Applications and Innovations in Intelligent Systems XVIII

BEST APPLICATION PAPER

Artificial Intelligence Techniques for the Berth Allocation and Container Stacking Problems in Container Terminals

Miguel A. Salido, Mario Rodriguez-Molins, Federico Barber[1]

Abstract The Container Stacking Problem and the Berth Allocation Problem are two important problems in maritime container terminal's management which are clearly related. Terminal operators normally demand all containers to be loaded into an incoming vessel should be ready and easily accessible in the terminal before vessel's arrival. Similarly, customers (i.e., vessel owners) expect prompt berthing of their vessels upon arrival. In this paper, we present an artificial intelligence based-integrated system to relate these problems. Firstly, we develop a metaheuristic algorithm for berth allocation which generates an optimized order of vessel to be served according to existing berth constraints. Secondly, we develop a domain-oriented heuristic planner for calculating the number of reshuffles needed to allocate containers in the appropriate place for a given berth ordering of vessels. By combining these optimized solutions, terminal operators can be assisted to decide the most appropriated solution in each particular case.

1 Introduction

Container terminals have become an important component of global logistics networks. The transshipment market is growing, although further analysis and development are needed to ensure reliability, delivery dates or handling times, increase productivity and container throughput from quay to landside and vice versa, etc. [1]. Several issues need optimization [2].

One of the main objectives in container terminals is to reduce the berthing time of vessels. This objective generates a set of interrelated problems mainly related to berth allocation, yard-side operation, storage operation and gatehouse operation. Usually, each one of these problems is managed independently of others due to

1 Instituto de Automatica e Informatica industrial, Universidad Politecnica de Valencia. Camino de vera s/n, Valencia, Spain {msalido,mrodriguez,fbarber}@dsic.upv.es

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_23,

their exponential complexity. However, these problems are clearly interrelated so that an optimized solution of one of them restrings the possibility of obtaining a good solution in another.

The overall goal collaboration between our group at the Technical University of Valencia (UPV) and the maritime container terminal MSC (Mediterranean Shipping Company S.A) is to offer assistance to help in planning and scheduling tasks such as the allocation of spaces to outbound containers, to identify bottlenecks, to determine the consequences of changes, to provide support in the resolution of incidents, to provide alternative berthing plans, etc.

In this paper, we focus our attention to the Berth Allocation Problem (BAP) and the Container Stacking Problem (CStackP) (see Figure 1). Briefly, the BAP consists on the allocation of docks (and cranes) to incoming vessels under several constraints and priorities (length and depth of vessels, number of containers, etc.). On the other hand, when a vessel berths, export containers stacked to be loaded in the vessel should be on top of the stacks of the container yard. Therefore, the CStackP consists on relocating the containers so that the yard crane does not need to do re-handling work at the time of loading. These two problems are clearly related: an optimal berth allocation plan may generate a large amount of relocations for export containers; meanwhile a suboptimal berth allocation plan could require fewer rearrangements. Terminal operators should decide which solution is the most appropriate in each scenario.

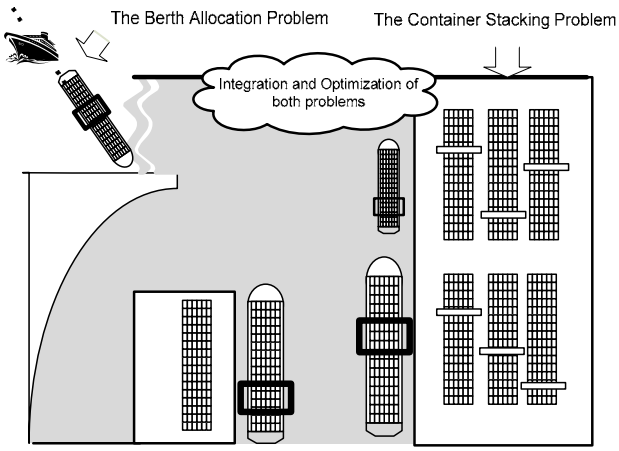


Figure 1. Integrated Remarshaling and Berthing problems in Maritime Terminals.

In this paper, we integrate a set of intelligent techniques for solving both problems concurrently in order to achieve a mixed-solution that combines optimization of both problems. To this end, we developed a heuristically-guided planner for generating a rehandling-free intra-block remarshaling plan for

container yards. Afterwards, we present a meta-heuristic approach for solving the BAP as an independent problem. Finally, we integrate optimization of both problems. Terminal operator should ultimately decide which solution is the most appropriate in relation to a multi-objective function: to minimize the waiting times of vessels and to minimize the amount of relocations of containers. These techniques will be very useful for terminal operators due to berth allocation is especially important in case of ship delays because in this case a new berthing place has to be allocated to the ship whereas containers are already stacked in the yard [2] and a remarshaling plan remains necessary to minimize the berthing time.

2 An Integrated Approach for Container Stacking and Berth Allocation Problems

As we have pointed out, both the CStackP and the BAP are well-known problems and several techniques have been developed to solve them separately. However, no systems have been developed to relate and optimize both problems in an integrated way. Only some works integrate the BAP with the Quay Crane Assignment Problem (QCAP), for instance [3] which follows to minimize the yard-related house-keeping costs generated by the flows of containers exchanged between vessels. However, there also exists a relationship between the optimization of maritime and terminal-sides operations (BAP, QCAP, CStackP, etc.).

Figure 2 shows an example of three berth allocation plans and a block of containers to be loaded in the vessels. Containers of type A, B and C must be loaded in vessels A, B and C, respectively. In the first berth allocation plan the order of vessel is A-B-C, the waiting time for this plan is 205 time units and the number of reshuffles needed to allocate the white containers at the top of the stacks is 110. The second berth allocation plan is B-A-C. In this case the waiting time for this plan is 245 time units and the number of reshuffles is 260. Finally, the third berth allocation plan is C-B-A, the waiting time for this plan is 139 time units and the number of reshuffles is 450. The question is straightforward: what is a better solution? A solution that optimizes the BAP problem could not be the more appropriate for the CStackP (and vice versa).

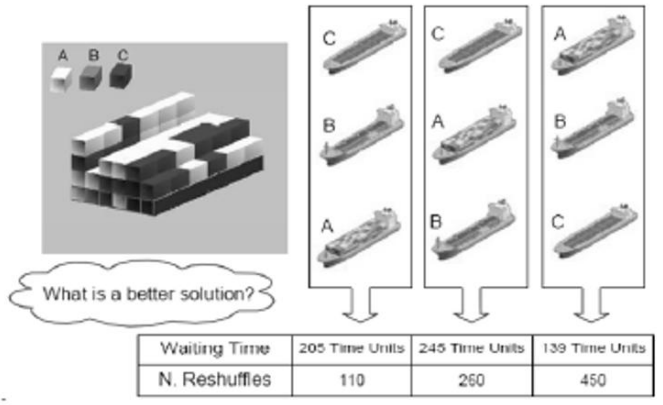


Figure 2. Three different plans for the BAP: What is better?

Given a waiting queue of vessels to be allocated and a given state of the containers in the container yard, each solution for the BAP ($SBAP_i$: a feasible sequence of mooring), requires a different number of container's re-locations in the associated CStackP solution ($SCStackP_i$) in order to put on top the containers to be loaded according to the order of berthing. We can associate a cost to each $SBAP_i$ related to the total weighted waiting time of vessels of this berthing order (T_w). Likewise, we can associate a cost to each $SCStackP_i$ as the number of required container relocations. Therefore, we can qualify the optimality of each global solution (Sol_i) of BAP and CStackP as a lineal combination of the quality of each partial solution:

$$Cost(Sol_i) = \alpha * Cost(SBAP_i) + \beta * Cost(SCStackP_i) \quad (1)$$

The best decision will depend on the policy of each maritime terminal (α and β parameters). The data flow diagram of the Integrated System Functioning can be seeing in Figure 3. Firstly, both the BAP and the CStackP data are loaded in the integrated system. Next, the BAP is solved to achieve a solution ($SBAP_i$) based on their constraints and criteria. Then, the CStackP is solved by taken into account the berthing order of vessels obtained in $SBAP_i$. The CStackP planner is applied sequentially for each vessel in $SBAP_i$, according the state of the container yard in each stage. Thus, the optimized remarshaling plan for the berthing order of vessels of $SBAP_i$ is obtained ($SCStackP_i$). After this step, the cost of the global solution (Sol_i) can be calculated by using the previous expression (1). By iterating this integrated process, the operators can obtain a qualification cost of each feasible Sol_i, as well as the best global solution (Sol_i), according the given α and β parameters. A branch and bound method has been also applied in the integrated

search of the best global solution (Sol_i), so that the search can be pruned each time the current solution does not improve the best solution found so far.

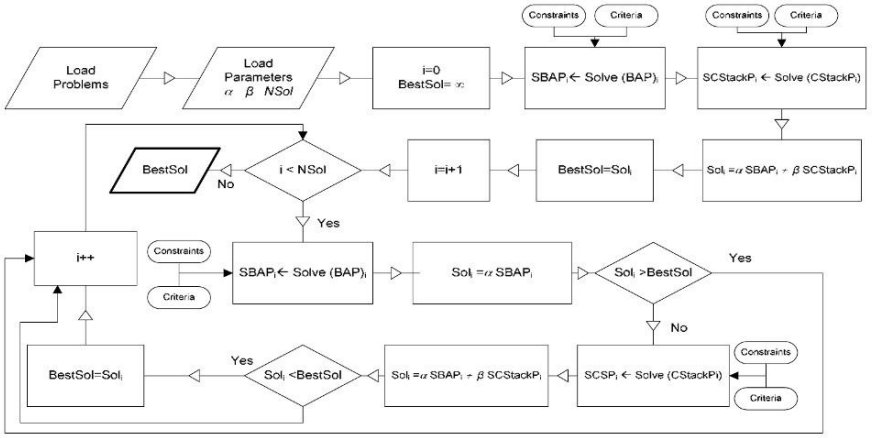


Figure 3. Data flow diagram of the Integrated System Functioning

In next sections we develop some techniques for solving the container stacking problem and the berth allocation problem in order to achieve a global solution Soli to the integrated problems.

3 A Domain-dependent Planner for the Container Stacking Problem

Containers are ISO standardized metal boxes which can be stacked on top of each other. A container yard is composed of several blocks, each one consisting of (20 - 30) yard-bays. Each yard-bay contains several (usually 6) rows and each row has a maximum allowed tier (usually 4 or 5 tiers for full containers).

Loading and offloading containers on the stack is performed by cranes following a 'last-in, first-out' (LIFO) criteria. Containers are stacked in the order they arrive. However, in the loading process of vessels, to access a container which is not at the top of its pile, those above it must be relocated. This remarshaling process is required since the stacking order depends on the order in which ships unload or containers have been stacked. This remarshaling process reduces the productivity of cranes and its optimization would minimize the moves required.

For safety reasons, it is usually prohibited to move the gantry crane while carrying a container [4], therefore these movements only take place in the same yard-bay. In addition, there exist a set of hard/soft constraints regarding container moves or locations where can be stacked, for example, small differences in height

of adjacent yard-bays, dangerous containers must be allocated separately by maintaining a minimum distance, etc.

The CStackP is a NP-complete combinatorial optimization problem and different approaches has been proposed [5,6]). The CStackP can be viewed, from the artificial intelligence point of view, as a modification of the Blocks World planning domain [7].

In [8], a planning system for remarshaling processes was proposed. This system obtains the optimized plan of reshuffles of containers in order to allocate all selected containers at the top of the stacks, or under another selected containers, in such a way that no reshuffles will be needed to load these outgoing containers.

The proposed planner was specified by means of the standard Planning Domain Definition Language (PDDL) [9] and it was developed on the well-known domain-independent planner *MetricFF* [10]. The developed domain file contains the common features of the problem domain: (i) the domain objects: containers and rows, (ii) the relations among them (propositions), and (iii) allowed moves to change the status of the problem (actions). The problem file describes each particular instance: (i) the initial layout of the containers in the yard (Initial state), (ii) the export containers (goal) which must be allocated at the top of the stacks or under other export containers, and (iii) the function to optimize (minimizing the number of relocation movements).

In [11] the Metric-FF-based initial planner was improved by integrating a domain-dependent heuristic (H1) in order to achieve efficiency. H1 computes an estimator of the number of container movements that must be carried out to reach a goal state, which it is used to guide search of solutions.

However, new constrains and optimization criteria are included in order to take into account real-world requirements:

1. Reducing distance of the out containers to the cargo side.
2. Increasing the range of the move actions set for the cranes allowing moving a container to 5th tier.
3. Balancing the number of stacked containers within the same bay in order to avoid sinks.

The improved planner can now manage a full container yard. The container yard is decomposed in yard-bays, so that the problem is distributed into a set of subproblems. Thus, each yard-bay generates a subproblem. However, containers of different yard-bays must satisfy a set of constraints among them. Therefore, subproblems are sequentially solved, so that each subproblem (yard-bay) takes into account the set of constraints with previously solved subproblems. This decomposition requires taking into account these new added constraints. With these new added constraint and criteria, the developed planner can solve more real-world based problems:

1. Balancing contiguous yard-bays: rows of adjacent yard-bays must be balanced in order to avoid sinks inter yard-bays (CB).

2. Dangerous containers must maintain a minimum security (Euclidian) distance among them (DC).

In order to insert our planner in the integrated system, we have improved our version to minimize the number of reshuffles for a set of out containers to be loaded in different vessels. Initially our planner was developed to minimize the number of reshuffles to allocate all goal containers at the top of the piles or under other goal containers. However, the order of the rest of containers in the yard-bay did not matter. The new planner takes into account these features and it is able to organize the bay in order to adapt to the berth schedule.

4 The Berth Allocation Problem

The BAP is one of the most relevant problems arising in the management of container ports. Several models are usually considered [12]:

- All vessels to be served are already in the port queue at the time that scheduling begins (static BAP).
- All vessels to be scheduled have not yet arrived but their arrival times are known (dynamic BAP)
- The quay is viewed as a finite set of berths, and each berth is described by fixed-length segments (Discrete BAP).
- Vessels can berth anywhere along the quay (Continuous BAP)

The objective in BAP is to obtain an optimal distribution of the docks and cranes to vessels waiting to berth. Thus, this problem could be considered as a special kind of machine scheduling problem, with specific constrains (length and depth of vessels, ensure a correct order for vessels that exchange containers, assuring departing times, etc.) and optimization criteria (priorities, minimization of waiting and staying times of vessels, satisfaction on order of berthing, minimizing cranes moves, degree of deviation from a pre-determined service priority, etc.).

The First-Come-First-Served (FCFS) rule can be used to obtain an upper bound of the function cost in BAP [13]. On the other hand, several methods have been proposed for solving BAP. Usually, these methods are based on heuristic [14] or metaheuristic [15,16] approaches. In [12], a comparative analysis is provided.

Our approach follows an integration of the Quay Crane Assignment Problem (QCAP) and the BAP through the metaheuristic Greedy Randomized Adaptive Search Procedure (GRASP) [17] which is able to obtain optimized solutions in a very efficient way. Following, we introduce the used notation:

$a(V_i)$	Arrival time of the vessel V_i at port.
$m(V_i)$	Moored time of V_i. All constraints must hold.
$c(V_i)$	Number of required movements to load and unload containers of V_i.
$q(V_i)$	Number of assigned Quay Cranes (QC) to V_i. The maximum number of

assigned QC by vessel depends on its length since a security distance is required (secQC). Let's assume that the number of QC does not vary along all the moored time. Thus, the handling time of V_i is given by (where *MovsQC* is the QC's moves per unit time):

$$\frac{c(V_i)}{q(V_i) \times MovsQC}$$

$d(V_i)$ Departure time of V_i, which depends on $m(V_i)$, $c(V_i)$, and $q(V_i)$.

$w(V_i)$ Waiting time of V_i from it arrives at port until it moors: $w(V_i) = m(V_i) - a(V_i)$.

$l(V_i)$ Length of V_i. There is a distance security between two moored ships: let's assume 5% of their lengths.

$pr(V_i)$ Vessels' priority.

In order to simplify the problem, let's assume that mooring and unmooring does not consume time and every vessel has a draft lower or equal than the quay. In each case, simultaneous berthing is allowed.

The goal of the BAP is to allocate each vessel according existing constraints and to minimize the total weighted waiting time of vessels:

$$T_w = \sum_i w(V_i)^\gamma \times pr(V_i)$$

The parameter γ ($\gamma \geq 1$) prevents lower priority vessels are systematically delayed. Note that this objective function is different to the classical tardiness concept in scheduling.

4.1 A meta-heuristic method for BAP

We have developed three different methods for solving BAPs. Firstly, we applied the simplest solution, following the FCFS criteria: $\forall i$, $m(V_i) \leq m(V_{i+1})$. A vessel can be allocated at time *t* when there is no vessel moored in the berth or there are available quay length and cranes at time *t* (Algorithm 1).

```
Data: V: set of ordered incoming vessels; b: state of the berth
Result: Sequence for V
V_last ← ∅;
V_m ← ∅;
foreach V_i ∈ V do
    t ← max(e(V_last), a(V_i));
    inst ← insertVessel(V_i, t, b);
    if !inst then
        T ← d(V_j)|V_j ∈ V_m ∧ d(V_j) > t;
        while t_k ∈ T∧!inst do
            inst ← insertVessel(V_i, t_k, b);
        end
    end
    update(b) ;    /* state of the berth b */
    V_last ← V_i;
    V_m ← V_m ∪ V_i;
end
```

Algorithm 1: Allocating vessels using FCFS policy

We also have implemented a complete search algorithm for obtain the best (optimal) mooring order of vessels: the lowest T_w (lower bound of the function cost). This algorithm uses the functions *moorVessel* (Algorithm 2) and *insertVessel* (Algorithm 3) to allocate one vessel from its arrival time (the required data are: v: Vessel for allocating; V_{in}: set of vessels already moored; b: state of the berth).

```
Data: v vessel to moor; V_in elements; b: state of
      the berth
inst ← insertVessel(v, a(v), b);
if !inst then
    T ← d(v_j)|v_j ∈ V_in ∧ d(v_j) > t;
    while t_k ∈ T∧ !inst do
        inst ← insertVessel(v, t_k, b);
    end
end
```

Algorithm 2. Function moorVessel. Allocating exactly one vessel in the berth.

However, with a complete search, only a limited number of vessels can be taken into account since search space grows exponentially. Therefore, we developed a meta-heuristic GRASP algorithm for berth allocation (Algorithm 4). This is a randomly-biased multistart method to obtain optimized solutions of hard combinatorial problems in a very efficient way. The parameter δ ($0 \leq \delta \leq 1$) allows tuning of search randomization.

```
Data: V_i: Vessel for allocating; t: actual time; b:
      state of the berth at time t;
Result: V_i could moor
if empty(b) then
    m(V_i) ← a(V_i);
    q(V_i) ← min(maxQC_Y, l(V_i)/secQC)
    d(V_i) ← m(V_i) + c(V_i) / (q(V_i) × movsQC);
    return true;
else
    freeQC ← QC(b) − Σ q(V_j)| t ≥ a(V_j) ∧ t < d(V_j);
    freeL ← l(b) − Σ l(V_j)| t ≥ a(V_j) ∧ t < d(V_j);
    if freeQC > 0 ∧ l(V_i) <= freeL then
        q(V_i) ← min(freeQC, l(V_i)/secQC)
        m(V_i) ← t;
        d(V_i) ← t + c(V_i) / (q(V_i) × movsQC);
        if checkResources(V_i) then
            return true;
        else
            return false;
        end
    else
        return false;
    end
end
```

Algorithm 3: Function insertVessel. Allocating one vessel in the berth at time t.

```
Data: δ factor; V_out elements; b: state of the berth
Result: Solution s
V_in ← {};
C ← V_out;
while C ≠ ∅ do
    foreach v_e ∈ V_out do
        f_c(v_e) ← 0;
        b' ← b;
        moorVessel(v_e, V_in, b');
        V'_in ← V_in ∪ v_e;
        foreach v_o ∈ V_out| v_o ≠ v_e do
            moorVessel(v_o, V'_in, b');
            f_c(v_e) ← f_c(v_e) + (w(v_o) * pr(v_o));
        end
    end
    c_inf ← min{f_c(e)| e ∈ C};
    c_sup ← max{f_c(e)| e ∈ C};
    RCL ← {e ∈ C|f_c(e) ≤ c_inf + δ(c_sup − c_inf)};
    v ← random(RCL);
    moorVessel(v, V_in, b);
    update(b) ;            /* state of the berth b */
    V_in ← V_in ∪ v;
    C ← C − v;
end
```

Algorithm 4. Allocating Vessels using GRASP metaheuristic

5 Evaluation

In this section, we evaluate the behaviour of the algorithms developed in the paper. The experiments were performed on random instances. For the CStackP, containers are randomly distributed in blocks of 20 yard-bays, each one with six stacks of 4 tiers. A random instance of a yard-bay is characterized by the tuple $<n,s>$, where '*n'* is the number of containers and *'s'* ($s \leq n$) is the number of selected containers in the yard-bay. A random instance for the BAP has 'k' vessels with an arrival exponential distribution with vessel's data randomly fixed (lengths, drafts, moves and priorities).

Table 1: Performance of real-world criteria in CStackPs

	Metric-FF Planner	**H1**	**CB**	**DC**	**CB+ DC**
Reshuffles	3.98	3.60	5.68	4.30	6.53
Sinks	24.33	32.67	0	33.33	0
Non-Safe Dangerous	15.33	7.67	8.00	0	0

For the developed planning system to solve CStackPs (Section 2), Table 1 shows the performance of the introduced real-world criteria. These experiments were performed on instances <15,4>. The results shown in Table 1 are the average of the best solutions found in 10 seconds and they represent the average number of reshuffles, the average number of sinks generated along the block, and the average number of unsatisfied dangerous containers. It can be observed that H1 outperforms the general purpose Metric-FF-based initial planner in the number of reshuffles and the new introduced criteria (CB, DC) avoid undesired situations.

Table 2 shows the computational times (in seconds) required for solving BAP by using a complete search against the GRASP method with 1000 iterations. As observed, complete search is impracticable from 12 vessels (more than 3 hours). However, the GRASP method takes around 30 seconds to solve a schedule of 20 vessels.

Table 2: Computing time elapsed (seconds) for BAP

No. Vessels	5	10	11	12	13	15	20
Complete search	< 1	112	110 5	118 30	574 62	-	-
Grasp	1	8	9	10	12	15	30

Table 3 shows the average waiting times using FCFS and Complete Search (CS) methods described for the BAP, with two different inter-arrival distributions (temporal separation among arriving vessels). Through these data, it is demonstrated that FCFS criteria results a schedule which is far away from the best one (CS).

Table 3: Total waiting time elapsed

No. Vessels	FCFS	CS
5 (separate arrival times)	73	46
10 (separate arrival times)	256	136
5 (closest arrival times)	117	80
10 (closest arrival times)	586	351

Using as minimization function the total weighted waiting time (T_w), Figure 4 shows the results given by the FCFS criteria, and the GRASP procedure (with 1000 iterations) respect to the value of δ. The optimum value is δ =0,3, which indicates the suitability of the cost function used in the GRASP procedure (Algorithm 4). A total of 20 vessels are allocated, with two different inter-arrival distributions (separate and closest arrival times) among them.

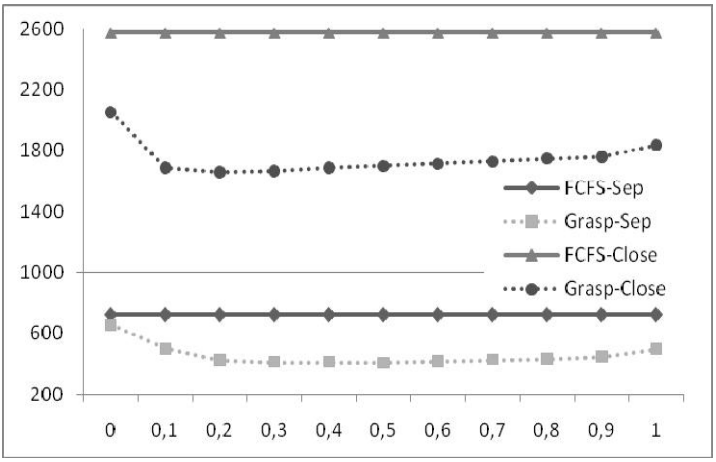


Figure 4: Weighted waiting time (Tw) with FCFS and GRASP procedures

As it was expected, the GRASP procedure obtains a lower T_w than the FCFS criteria. It is also remarkable that using GRASP is more profitable when the inter-arrival distribution of the vessels is closer. It is not possible to know the optimal T_w due to the exponential computational time required by a complete search with 20 vessels.

Finally, Figure 5 shows the combined function cost $Cost(Sol_i)$, introduced in (1) which relates: (i) The normalized total weighted waiting time of vessels, $Cost(SBAP_i)$, and (ii) the number of its required container relocations, $Cost(SCStackP_i)$; for ten different scenarios. In each one of this ten cases, the arrival times and data of vessels, as well as the initial state of the container yard, have been randomly generated. Figure 5 represents the combined function cost, $Cost(Sol_i)$ with three different weights of the parameters α and β. We can see that

better (or worst) berthing orders can require larger (or smaller) number of container relocations.

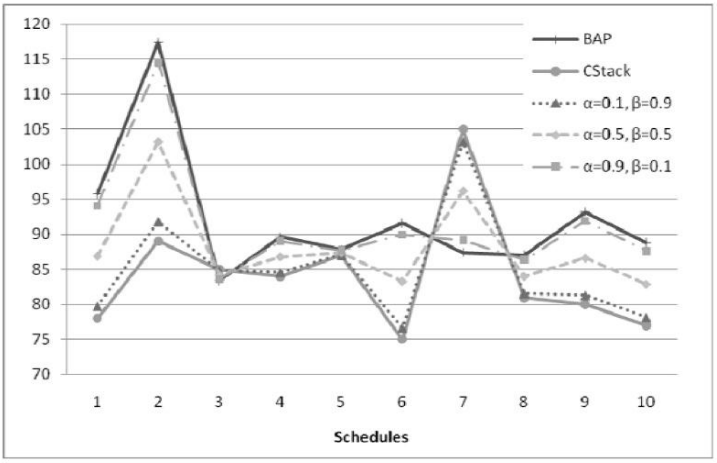


Figure 5. Relating the costs of BAP and CStackP

6 Conclusions

This paper presents an improved planning system for obtaining optimized plans for remarshaling processes required in CStackPs. A multi-start GRASP method has been also developed for obtaining optimized berthing orders in BAPs. Several evaluations on randomized scenarios have been performed. The conclusion is clear and expected: a better ordering of berthing of vessels can imply a higher number of container's relocations, in order to release the containers according to the order they should be loaded on vessels; and vice versa. This implies a multi-criteria decision. As future work, we are working on a function to estimate the number of reshuffles needed for each berth plan without needing to execute the planner. Furthermore we plan to improve the GRASP method and adequate the parameters (α, β and γ) to real-world practical decisions and expert knowledge. Then, the developed system, as a computer-based aid system, could assist container terminal's operator to simulate, evaluate and compare different feasible alternatives.

Acknowledgments

This work has been partially supported by the research projects TIN2007-67943-C02-01 (MEC, Spain-FEDER), and P19/08 (M. Fomento, Spain-FEDER).

References

1. Henesey, L. (2006). Overview of Transshipment Operations and Simulation. In: MedTrade conference, Malta, April. pp. 6–7.
2. Stahlbock, R. and Voß, S. (2008). Operations research at container terminals: a literature update. OR Spectrum 30(1), 1–52.
3. Giallombardo, G., Moccia, L., Salani, M., and Vacca, I. (2010). Modeling and solving the tactical berth allocation problem. Transportation Research Part B: Methodological 44(2), 232–245.
4. Yusin, L., and Hsu, N.Y. (2007). An optimization model for the container pre-marshalling problem. Computers & Operations Research 34(11), 3295 – 3313.
5. Park, K., T. Park and K.R. Ryu (2009). Planning for remarshaling in an automated container terminal using cooperative coevolutionary algorithms. In: ACM symposium on Applied Computing. ACM. pp. 1098–1105.
6. Kim, K.H. and Hong G.P. (2006). A heuristic rule for relocating blocks. Computers & Operations Research 33(4), 940–954.
7. Winograd T. (1971). Procedures as a representation for data in a computer program for understanding natural language. MIT. Cent. Space Res.
8. Salido, M., Sapena, O and Barber F. (2009). The Container Stacking Problem: an Artificial Intelligence Planning-Based Approach. In Proc. of The Int. Workshop on Harbour, Maritime and Multimodal Logistics Modelling and Simulation HMS'2009. pp:127-131.
9. Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). PDDL - the planning domain definition language. AIPS-98 Planning Committee.
10. Hoffmann, J. (2003). The metric-FF planning system: translating "ignoring delete lists" to numeric state variables. J. Artif. Int. Res. 20(1), 291–341.
11. Rodriguez, M, Salido, M., Barber F. (2009a). Domain-Dependent Planning Heuristics for Locating Containers in Maritime Terminals. Trends in Applied Intelligent Systems. IEA/AIE 2010, LNAI 6096, pp. 742–751.
12. Theofanis, S., Boile, M. and Golias M.M. (2009). Container terminal berth planning. Transportation Research Record: Journal of the Transportation Research Board 2100(-1), 22–28.
13. Lai, KK and Shih, K. (1992). A study of container berth allocation. Journal of Advanced Transportation 26(1), 45–60.
14. Guan, Y. and Cheung, R.K. (2004). The berth allocation problem: models and solution methods. OR Spectrum 26(1), 75–92.
15. Cordeau, J.F., Laporte, G., Legato, P. and Moccia, L. (2005). Models and tabu search heuristics for the berth-allocation problem. Transportation science 39(4), 526–538.
16. Cheong, C.Y., Tan, K.C. and Liu, D.K. (2009). Solving the berth allocation problem with service priority via multi- objective optimization. In: Computational Intell. in Scheduling, 2009. CI-Sched '09. IEEE Symposium on. pp. 95 –102.
17. Feo, T.A. and Resende, M.G.C. (1995). Greedy randomized adaptive search procedures. Journal of Global Optimization 6(2), 109–133.

APPLICATIONS OF MACHINE LEARNING I

Social Network Trend Analysis Using Frequent Pattern Mining and Self Organizing Maps

Puteri N. E. Nohuddin[1], Rob Christley[2], Frans Coenen[1], Yogesh Patel[1,3], Christian Setzkorn[2], Shane Williams[3]

Abstract A technique for identifying, grouping and analysing trends in social networks is described. The trends of interest are defined in terms of sequences of *support values* for specific patterns that appear across a given social network. The trends are grouped using a SOM technique so that similar tends are clustered together. A cluster analysis technique is then applied to identify "interesting" trends. The focus of the paper is the Cattle Tracing System (CTS) database in operation in Great Britain, and this is therefore the focus of the evaluation. However, to illustrate the wider applicability of the trend mining technique, experiments using a more standard, car insurance, temporal database are also described.

1 Introduction

Social network mining is a popular area of study. The aim is to extract knowledge from such networks. The networks themseves are conceptualised as graphs comprising nodes and links. Common examples of social networks are www applications such as Facebook, Bebo and Flicker. However, in the wider context social networks can include business communities, file sharing systems and co-authoring frameworks. In each case, the nodes represent individuals and the links communications. These communications often take the form of text (emails) but can be files (photopgraphs, movies, etc.). In this paper, we have abstracted out the concept of social networks even further to encompass the Cattle Tracing System (CTS) in operation in Greart Britain (GB). CTS incorporates a database that records cattle movements. The CTS database can be viewed as a large scale social network where the nodes

1 Department of Computer Science, University of Liverpool, UK,
puteri, frans@liverpool.ac.uk

2 School of Veterinary Science, University of Liverpool and National Centre for Zoonosis Research, Leahurst, Neston, UK,
robc, c.setzkorn@liverpool.ac.uk

3 Deeside Insurance Ltd., Deeside, UK,
yogesh, shane@deesideinsurance.co.uk

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_24,

represent cattle holding areas (farms, markets, abattoirs, etc) and the links are cattle movements between locations. This *cattle movement* social network can be mined, using relatively standard network mining techniques, to find (say) clusters of nodes. However, in this paper, the authors are interested in the dynamic mining of such social networks, as opposed to their static mining. The authors are particularly interested in mechanisms for identifying trends in social network in general, and the cattle movement social network in particular. The objective is to identify trends and variations in these trends. In the context of the cattle movement social network, the identification of trends and change points will provide knowledge of (say) the effect of the introduction of new legislation, or indicate changes in working practices, it will also give an insight into the way that cattle infections may spread through GB.

The trends we are interested in are defined as the changing frequencies with which common patterns occur across social network data. Trends are collected according to *epochs*, which can then be compared. The nature (duration) of an epoch is application dependent. However, for the cattle movement social network, it made sense to consider trends in terms of years (i.e. the duration of an epoch is 12 months) because this will serve to capture seasonal variations. Whatever the case, the epoch length is a user supplied variable that can be easily adjusted to fit alternative applications.

Using the proposed trend mining mechanism, a significant number of trends may be identified, too many to allow simple inspection by decision makers. Some mechanism was therefore required to allow the simple presentation of trend lines. The first technique advocated in this paper is to group (cluster) trends that display similar contours. To this end, Self Organising Map (SOM) technology has been adopted. Once the trends have been identified and grouped, we wish to determine how these trends change from epoch to epoch. The nature of the changes which we might be interested in will vary from application to application. For some applications, we may be interested in trends that remain constant, for others we may be interested in trends that change radically. To identify changes in trends, the advocated approach is to generate a sequence of SOM maps, one per epoch, and analyse how trends "move" (or do not move) from SOM to SOM (epoch to epoch).

The proposed approach to *trend mining*, using a sequence of SOMs has much wider application. Thus, although the focus of this paper is the cattle movement social network, for evaluation purposes, we also consider an alternative application, namely that of a customer database. More specifically a car insurance database containing requests for insurance quotes from potential customers. The "network" in this case is much simpler in that the nodes represent geographical locations which all communicate with a central "broker" node. The links in this case were labelled with the amount of traffic per time stamp (instead of the number of cattle moved per time stamp).

The contribution of this paper may thus be summarised as: (i) an unusual application of social network mining with respect to the CTS database, (ii) a mechanism for generating frequent pattern trends, (iii) a process for assisting the analysis of the identified trends using SOM technology and (iv) an approach to identify "interesting" changes in trends. The rest of this paper is organised as follows. Some previous

work is described in Section 2. The proposed social network trend mining approach is described in Section 3. Sections 4 and 5 present an evaluation of the proposed technique, firstly using the cattle movement social network (the focus of this paper), and secondly a car insurance time stamped data set to illustrate the wider application of the proposed technique. Some conclusions are then presented in Section 6.

2 Previous Work

The general availability of advanced computer information systems have resulted in the rapid growth of temporal databases together with a corresponding desire to identify (mine) trends in these collections. For example, Google Trends, a public web facility that supports the identification of trends associated with keyword search volume [1]. Trend recognition processes can be applied to both qualitative and quantitative data, such as the forecasting of financial market trends based on numeric financial data, and usage of text corpi in business news [2]. Raza and Liyanage [4] proposed a trend analysis approach to mine and monitor data for abnormalities and faults in industrial production. There are many more examples, however, in this paper, we are interested in mining trends which are defined in terms of the changing frequency of individual patterns presented in the data.

A social network depicts the structure of some social entity, and normally comprises actors who are connected through one of more links [18]. To analyze this structure, techniques have been proposed which map and measure the relationships and flows between nodes. Social network mining can be applied in a static context, which ignores the temporal aspects of the network; or in a dynamic context, which takes temporal aspects into consideration. In a static context, we typically wish to: (i) find patterns that exist across the network, (ii) cluster (group) subsets of the networks, or (iii) build classifiers to categorize nodes and links. In the dynamic context, we wish to identify relationships between the nodes in the network by evaluating the spatio-temporal co-occurrences of events [6]. The latter is thus the focus of the work described in this paper.

As noted above, in this work, we define trends in terms of the changing frequency of patterns with time. A frequent pattern, as first defined by Agrawal et al. [7], is a subset of attributes that frequently co-occur according to some user specified support threshold. The frequent pattern idea has been extended in many directions. A number of authors have considered the nature of frequent patterns with respect to the temporal dimension, for example sequential patterns [8], frequent episodes [9], emerging patterns [10] and jumping and emerging patterns [3]. Many alternative frequent pattern mining algorithms, that seek to improve on Agrawal's original Apriori algorithm, have also been proposed. TFP (Total from Partial) [11] is one established algorithm that extends Apriori. For the work described here, TFP has been adapted for the purpose of trend mining.

Self Organising Maps (SOMs) were first introduced by Kohonen [13, 12]. Fundamentally, SOMs are a neural network based technique designed to reduce the number of data dimensions in some input space by projecting it onto a $n \times m$ "node

map", which plots the similarities of the input data by grouping (clustering) similar data items together at nodes. The SOM learning process is unsupervised, in other words no predefined number of clusters is specified. Currently, there is no scientific method for determining the best values for $n \times m$, i.e. to identify how many clusters should be represented by a SOM. However, the $n \times m$ value does define a maximum number of clusters; although on completion some nodes may be empty [16]. Since SOM are based on competitive learning, the output nodes on the map compete among each other to be stimulated to represent the input data. With respect to the work described in this paper, we have adopted a SOM approach to group similar trends, and thus provide a mechanism for analysing social network trend mining results.

For many applications, such as those considered in this paper, we are interested in detecting changes in trends. This can be achieved by applying *cluster analysis* techniques to the SOM generated maps. Cluster analysis is concerned with the discovery of information about the relationship and/or similarity between clusters. When conducting cluster analysis practitioners are predominantly interested in cluster size enlargement and reduction and cluster membership migration. Several methods have been introduced to detect cluster changes and cluster membership migration. For example, Lingras et. al. [19] proposed the use of Temporal Cluster Migration Matrices (TCMMs) to visualize cluster changes in e-commerce sites usage that reflected changes in user spending patterns. A simple Euclidean distance measure is adopted in this paper.

3 The Trend Mining Mechanism

As noted in Section 1, a trend is defined as a sequence of *support* values, associated with a specific pattern, over a sequence of time stamps. The support of a pattern is the number of occurences of that pattern in the data set for some time stamp. The sequence of time stamps is referred to as an *epoch*. Thus, a trend t comprises a set of values $\{v_1, v_2, \ldots, v_n\}$ where n is the number of time stamps in the epoch. A trend associated with a particular pattern i is indicated by t_i. The jth value in a trend t_i is indicated by t_{ij}. We wish to identify changes in the trends associated with individual patterns and thus we wish to compare trends over two or more epochs. A sequence of trends T comprises a set of trends $\{t_1, t_2, \ldots, t_e\}$, where e is the number of epochs described by the sequence. The proposed approach (Figure 1) comprises three stages: (i) frequent pattern trend mining, (ii) trend clustering, and (iii) analysis of trend clusters.

3.1 Frequent Pattern Trend Mining

The input to the trend mining system comprises a binary valued, time stamped, data set $D = \{d_1, d_2, \ldots, d_{n \times e}\}$ (recall that n is the number of time stamps per epoch, and e

is the number of epochs under consideration). The records in each dataset in D comprise some subsets of a global set of binary valued attributes $A = \{a_1, a_2, \ldots, a_m\}$. The number of records in each data set need not be constant across the collection. The patterns we are interested in are thus also subsets of A. To limit the overall number of patterns a *support threshold* is used, in the same way as in Association Rule Mining (ARM). A pattern is not deemed to be "interesting" unless its number of occurrences in an individual dataset d is greater than this threshold. Some examples patterns are given in the Figure 1. Thus, the pattern {a,b,c,d} has a sequence of support values of {0, 0, 2500, 3311, 2718, 0, 0, 0, 2779} describing a nine time-stamp trend associated with a single epoch, similar sequences may be extracted for all e epochs. Note that a 0 support value indicates a support value below the support threshold.

To mine the trends, an extended version of the TFP algorithm [11] was used. TFP is an established frequent pattern mining algorithm distinguished by its use of two data structures: (i) a P-tree to encapsulate the input data and conduct a partial pattern count in the process, and (ii) a T-tree to store identified patterns. The T-tree is essentially a reverse *set enumeration tree* that allows fast *look up*. The TFP algorithm, in its original form, was not designed to address the temporal aspect of frequent pattern mining. The algorithm was therefore extended so that a sequence of $n \times e$ data sets could be processed and the frequent patterns stored in a way that would allow for differentiation between individual time stamps and epochs. The resulting algorithm was called TM-TFP (Trend Mining TFP) which incoporated a

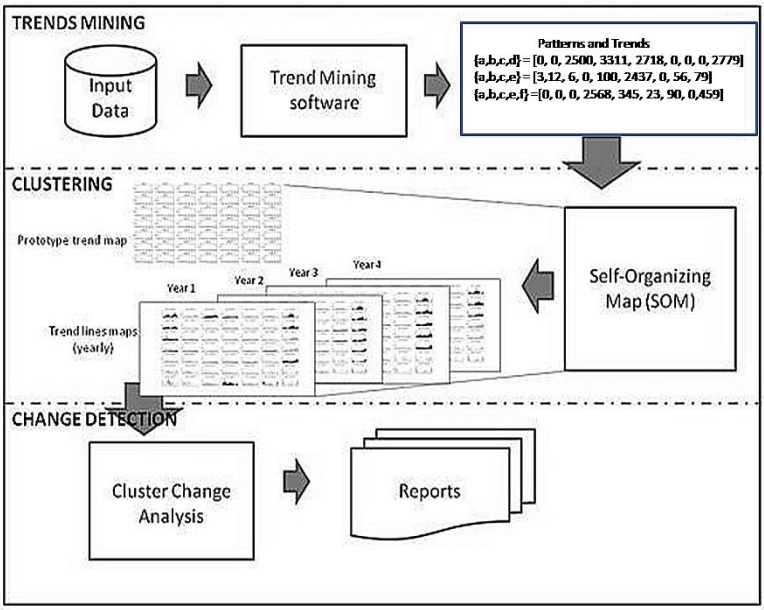


Fig. 1 Trend Mining Framework

TM-T-tree to store the desired patterns. An overview of TM-TFP is given in Figure 2. The *buildTM_Tree* method processes the collection of T-trees built from the input data sets. The *addToTMtree* method adds an item set node to the TM T-tree with its support value. The resulting trends are the input data for the clustering process which is described in the following subsection.

3.2 Trend Clustering

The process described above for identifying trends operates successfully, but produces a great many trend lines when a low support threshold is used (the option to use a higher threshold does reduce the number of trends but entails the risk of missing potentially interesting trends). The large number of trends produced makes it difficult for decision makers to interpret the result. Some mechanism for assisting the desired interpretation was therefore desirable. The idea of clustering similar trends allows decision makers to focus on particular groups of trends. The concept of clustering is well established in the data mining community, however little work has been directed at clustering time series (trend lines). The approach advocated in this paper is to use Self Organising Maps (SOMs). Using the SOM concept one map was created per epoch. The SOM was initialized with $n \times m$ nodes such that each

```
buildTM_Ttree() {
sizeTMtree =
TMtfpObj[0].getNumOneItemSets();
for (i=0;i<TMtfpObj.length;i++)
       size =
TMtfpObj[i].getNumOneItemSets();
       if (size>sizeTMtree)
sizeTMtree=size;
       startTM_TtreeRef=new
TM_TtreeNode[sizeTMtree+1];
for (i=1;i<startTM_TtreeRef.length;i++)
       startTM_TtreeRef[i] = new
       TM_TtreeNode(numofDataSets);
       for (i=0;i<TMtfpObj.length;i++)
       buildTM_Ttree(TMtfpObj[i],i);
}
buildTM_Ttree(PartialSupportTree
linkRef, index) {
numOneItemSets=
   linkRef.getNumOneItemSets();
TtreeNode[] tree =
linkRef.getStartOfTtree();
for (int i=1;i<=numOneItemSets;i++)
if (tree[i]!=null)sup = tree[i].support;
if (sup >= minSupport)
       itemSet = newshort[1];
       itemSet[0] = (short) i;
       addToTMtree(itemSet,sup,index);
Move down a level
if (tree[i].childRef!=null)
       buildTM_Ttree(itemSet, i,
       tree[i].childRef,index);
}
```

```
buildTM_Ttree(itemSetSofar,size,
      TtreeNode[] tree,index) {
for (i=1;i<size;i++)
if (tree[i] != null)
sup = tree[i].support;
if (sup >= minSupport)
itemSet=newshort[itemSetSofar.length+1];
itemSet[0]=(short) i;
for (j=1;j<itemSet.length;j++)
     itemSet[j]=itemSetSofar[j-1];
addToTMtree(itemSet,sup,index);
        Move down a level
       if (tree[i].childRef!=null)
       buildTM_Ttree(itemSet,i,
}
addToTMtree(itemSet, support,index) {
endIndex = itemSet.length-1;
      addToTMtree(startTM_TtreeRef,
      startTM_TtreeRef.length+1,
endIndex,itemSet,support,index);
}
addToTMtree(linkRef,size,endIndex,itemSet,
            support, index) {
if (linkRef == null)
linkRef = new TM_TtreeNode[size];
for(i=1;i<linkRef.length;i++)
linkRef[i] = null;
currentAttribute = itemSet[endIndex];
if (linkRef[currentAttribute] == null)
linkRef[currentAttribute]=
       new TM_TtreeNode(numofDataSets,
         index,support);
if (endIndex == 0)
linkRef[currentAttribute].addSupportValue(i
ndex,support);
return(linkRef);
linkRef[currentAttribute].childRef
addToTMtree(linkRef[currentAttribute].
       childRef, currentAttribute,endIndex-
       1,itemSet,support,index);
return(linkRef);
}
```

Fig. 2 TM-TFP Algorithm

node represented a category of trend; the map was then trained and the remaining examples assigned to nodes using a distance function. The authors experimented with different mechanisms for training the SOM, including: (i) devising specific trends to be represented by individual nodes, (ii) generating a collection of all trends that are arithmetically possible and training the SOM using this set and (iii) using some or all of the trends in the first epoch to be considered. The first required prior knowledge of the trend configurations in which we might be interested. It was discovered that the second resulted in a map for which the majority of nodes were empty. The third option was therefore adopted, the SOM was trained using the trend lines associated with the first epoch. The resulting *prototype map* was then populated with data for all e epochs to produce a sequence of e maps. Figure 3 outlines the basic SOM algorithm.

3.3 Analysis of Trend Clusters

Change points in trend analysis can be interpreted in a number of ways. At its simplest, they may be interpreted as an abrupt change in direction of a trend line. A more complex interpretation may be the existence of changes in amplitude and/or frequency of fluctuating (seasonal) trends. Alternatively, an end user may be interested in an absence of change points. The interpretation applied to the cattle movement database is that we are interested in trends, associated with particular patterns, that change from epoch to epoch, i.e. are not consistent across the sampled temporal range. To this end, a simple cluster analysis technique was applied to identify trends that change location in the SOM associated with one epoch to the SOM associated with a subsequent epoch. The change can be measured by translating the trend line maps into a rectangular (D-plane) sets of coordinates and applying a Manhattan or Euclidean distance function to observe the similarities and differences of trends across the epochs. The greater the distance moved, the more significant the change.

```
t = current iteration
I = limit on time iteration
w = current weight vector
x = target data input
y = yearGenerate

Prototype vector map from data input of y
•Randomize the map's nodes' weight vectors, w.
•Grab an input vector, x.
•Traverse each node in the map
      •Use Euclidean distance formula to find similarity between the input
      vector and the map's node's weight vector
      •Track the node that produces the smallest distance (this node is the
      best matching unit, BMU)
•Adaptive process which updates the nodes in the neighbourhood of BMU by
pulling them closer to the input vector.
•Increment t and repeat from 2 while t < I

Generate Trend line maps
•Load labels (data input) of y
•For all data input, fit to BMU nodes on Prototype map
•Increment y and repeat from 1
```

Fig. 3 Basic SOM Algorithm

Thus, given a sequence of trend-line maps (SOMs) comparisons can be made to see how trends associated with individual frequent patterns change by analyzing the nodes in which they appear. The process may be described as follows:

1. Generate a matrix measuring $e \times k$ (k = number of frequent patterns).
2. Populate matrix with the node number for each pattern per *epoch*.
3. Calculate distance moved and store.
4. Identify movements above a given threshold.

4 Experimental Analysis Using The Cattle Movement Social Network

This, and the following, section presents an experimental analysis of the above described approach to trend mining. This section is directed at the cattle movement database which has provided the central focus of the work described. The following section considers a customer "database" so as to determine the potential benefits of the wider application of the proposed technique. This section commences with an overview of the cattle movement database and its transformation into a social network, followed by an analysis of the trend mining process as a applied to the generated network.

4.1 Cattle Movement Database

The CTS database records all the movements of cattle registered within or imported into GB. The database is maintained by the Department for Environment, Food and Rural Affairs (DEFRA). Cattle movements can be "one-of" movements to final destinations, or movements between intermediate locations. Movement types include: (i) cattle imports, (ii) movements between locations, (iii) movements in terms of births and (iv) movements in terms of deaths. The CTS was introduced in September 1998, and updated in 2001 to support disease control activities. Currently, the CTS database holds some 155 Gb of data.

The CTS database comprises a number of tables, the most significant of which are the animal, location and movement tables. For the analysis reported here the data from 2003 to 2006 was extracted to form 4 epochs each comprising 12 (one month time stamps). The data was stored in a single data warehouse such that each record represented a single cattle movement instance associated with a particular year (epoch) and month (time stamp). The number of CTS records represented in each epoch was about 400,000. Each record in the warehouse comprised: (i) a time stamp (month and year), (ii) the number of cattle moved, (iii) the breed, (iv) the senders location in terms of easting and northing grid values, (v) the "type" of the sender's location, (vi) the receivers location in terms of easting and northing grid values, and (vii) the "type" of the receiver's location. If two different breeds of

cattle were moved at the same time from the same sender location to the same receiver location, this would generate two records in the warehouse. The maximum number of cattle moved between any pair of locations for a single time stamp was approximately 40 animals.

4.2 Cattle Movement Trend Mining

The TM-TFP algorithm was applied to the cattle movement social network and frequent pattern trends generated. For experimental purposes three support threshold values, 0.5%, 0.8% and 1.0% were used. Table 1 presents the number of frequent patterns trends discovered for each of the 4 epochs using the three support thresholds. As expected, the lower the support threshold used the greater the number of generated trends. Note also that the number of trends increases exponentially. An example of the nature of a frequent pattern, in the context of the cattle movement social network, is:

$$\{numberAnimalsMoved \leq 5, SenderPTI = 4, ReceiverArea = 54, \\ SenderLocationType = Agricultural\ Holding, SenderArea = 53, \\ AnimalAge \leq 1 year\ old\}$$

(the values for the *ReceiverArea* and *SenderArea* are Ordinance Survey grid square numbers). The associated sequence of support values (for 2003) representing the trend line for that year were:

$$[2391, 2609, 3218, 3009, 3890, 2759, 2298, 3124, 2911, 3331, 3791, 2417]$$

Table 1 Number of frequent pattern trends identified using TM-TFP for sequence of four cattle movement social network epochs and a range of support thresholds

Year	Support Threshold		
	0.5%	0.8%	1%
2003	63,117	34,858	25,738
2004	66,870	36,489	27,055
2005	65,154	35,626	25,954
2006	62,713	33,795	24,740

The generated trends were clustered using the SOM technique. The SOM was initializing with 7×7 nodes, and trained using the frequent pattern trends produced for the (earliest) 2003 year. The resulting prototype map is shown in Figure 4. Inspection of this map shows, for example, that node 1 (top-left) represents trend lines associated with patterns with higher support in spring (March to May) and autumn (September to November). Alternatively, node 43 (bottom-left) indicates trend lines with high support in spring only (March to April). Note that the distance between nodes indicates the dissimilarity between them; the greatest dissimilarity is thus between nodes at opposite ends of the diagonals. Once the initial prototype map has been generated, a sequence of trend line maps can be produced, one for each epoch.

Figure 5 gives the 2003 map. Note that in Figure 5, each node has been annotated with the number of trends in the "cluster" and that the "darker" trend lines indicate a greater number of trend lines within that cluster.

The cluster analysis mechanism highlighted interesting information beneficial to decision makers. Table 2 shows some example trends (representing frequent patterns) migrate from one cluster to another. Thus, the trend line representing the pattern $\{numberAnimalsMoved \leq 5, ReceiverPTI = NULL, ReceiverLocationType = Calf\ Collection\ Centre, SenderLocationType = Agricultural\ Holding, SenderArea = 14, AnimalAge \leq 1 year\ old, Gender = female\}$ was in node 49 (bottom right in Figure 5) in 2003 and 2004, but then migrated to node 48 in 2005 and disappeared in 2006.

Table 2 Examples of CTS Frequent Patterns migrating from one SOM node to another

Frequent Pattern Code	Node 2003	Dist	Node 2004	Dist	Node 2005	Dist	Node 2006
{441 436 329 301 213 4 3}	49	0	49	1	48	0	0
{441 436 329 301 213 196}	48	1	49	4.1	38	3.2	48
{378 301 263}	39	0	39	3.2	49	3.2	39
{441 329 214}	47	2	49	0	0	0	49

Using the above cluster analysis technique decision makers can "focus in" on particular types (clusters) of trends. In terms of further reducing the overall number of trend lines this can be achieved by considering only a subset of the detected frequent patterns according to particular attributes of interest. The term *meta-pattern* is introduced to represents a way of considering groups of patterns. In the context

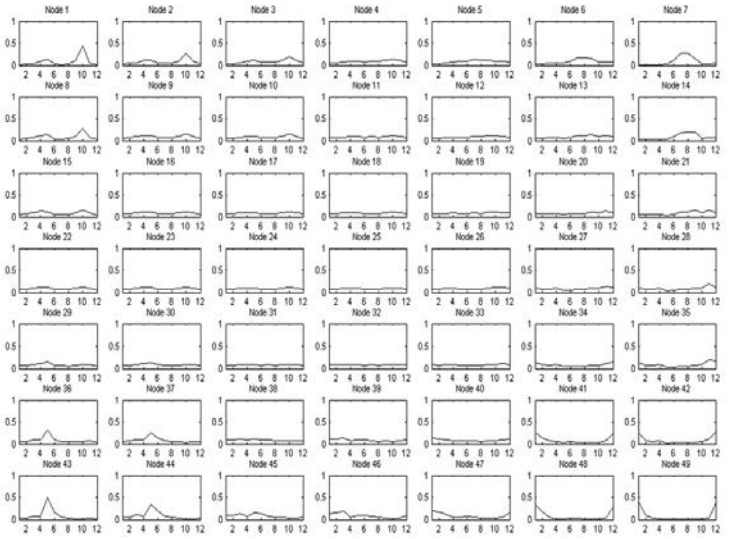


Fig. 4 CTS prototype map

of the cattle movement social network, we are interested in patterns that include spatial information (i.e. sender and receiver locations). Four categories of meta-pattern were therefore identified:

1. **Movement from start points**: patterns that include movement and sender attributes/columns.
2. **Movement to end points**: patterns that include movement and receiver attributes/columns.
3. **Movement from start to end points**: patterns that include movement and both sender and receiver attributes/columns.
4. **Movement for other non spatial attributes**: patterns which do not feature the above.

Meta-patterns form smaller groups of patterns for cluster and trend analysis thus simplifying the cluster analysis task.

5 Car Insurance Trend Mining

The above described technique has application with respect to alternative types of data. For example, if we consider a standard time stamped, tabular data set, we can identify trends in this data in the same manner as described for the cattle movement social network. This is illustrated in this section by considering a car insurance quote data set, the Deeside data set [1]. The data can be viewed as representing a "star" net-

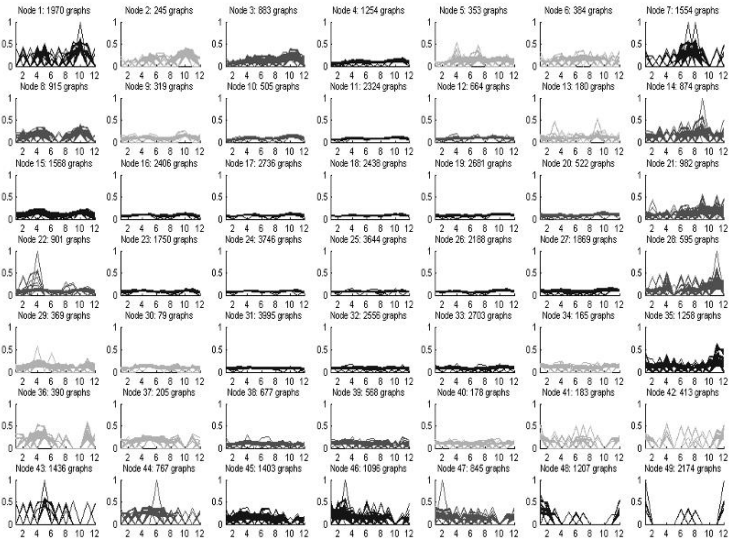


Fig. 5 CTS Map for 2003 frequent pattern trends

[1] The data set was provided by Deeside Insurance Ltd, Deeside, UK.

work with Deeside at the center as a *super node* and all other nodes radiating out from it. The outlying nodes represent geographical locations defined by the first characters of customer postcodes. The links are labelled with the number of interconnections between individual geographic locations and the center. The data set was partitioned into monthly time stamps and two epochs (2008 and 2009). Each month comprises some 1000 records. Each record consists of 13 attributes: (i) Aggregator [2], (ii) year of insurance contract, (iii) customer gender, (iv) make of car, (v) car engine size, (vi) year of manufacture, (vii) customer postcode, (viii) driver age (ix) conviction code, (x) conviction code number (xi) length of disqualification, (xii) fault and (xiii) penalty (note that the value for some of the attributes is *null*).

Table 3 Number of frequent pattern trends identified using TM-TFP for sequence of two Deeside Insurance epochs and a range of support thresholds

Year	Support Threshold		
	2%	3%	5%
2008	314,471	142,175	55,241
2009	284,871	122,371	49,983

Table 3 presents the number of trends generated by applying TM-TFP to the Deeside data set using a range of support thresholds of 2%, 3% and 5% respectively. Note that lower support thresholds were used than in the case of the CTS dataset because the Deeside data was smaller. The results presented in Table 3 corroborate those presented previously in Table 1. An example of a frequent pattern found in the Deeside data is:

$$\{Fault = NoBlame, LengthOfDisqualify \leq 5, Age \leq 50, \\ PostCodeArea = CH, CustomerGender = female\}$$

The associated sequence of trend line values (for 2008) were:

$$[23, 0, 31, 18, 0, 4, 0, 7, 25, 9, 16, 19]$$

A 7×7 SOM was again used and trained using the 2008 data. The prototype map is presented in Figure 6. From the figure it can be seen, for example, that node 1 indicates a trend line with high support mainly in February, whilst node 7 shows a trend line with high support mainly in March. It is interesting to note that there are more identified patterns in the first and last quarters of the year. The prototype map was then populated with the 2008 and 2009 data to produce a sequence of two maps that could be compared. Comparison of clusters allowed for the identification of changes in customer "quote request" habits. Table 4 presents some examples of trend migrations identified from within the Deeside Insurance data set. For example, the trend line representing the pattern $\{310, 286, 283, 145\}$ which translates to $\{Fine \leq 1000, ConvictCode = SP, 41 \leq DriverAge \leq 50, 1996 \leq CarYearManufacture \leq 2000\}$ which was in node 43 (bottom right in Figure 6) in

[2] An aggregator is a web application or search facility that allows users to obtain and compare a number of insurance quotes/prices.

2008 migrated to node 11 in 2009. This signifies that the pattern has changed from a trend with high support in September to the trend with high support in February and March.

Table 4 Example of Deeside Insurance Frequent Patterns that migrated to other clusters

Frequent Patterns	Node 2008	Dist	Node 2009
{310 286 283 145}	43	5.8	11
{310 286 283 145 1}	44	6.3	4
{310 286 283 146}	36	4.2	18
{310 286 283 146 1}	35	2.2	20

6 Conclusion

A social network trend mining mechanism has been described, founded on frequent pattern mining, SOM clustering and cluster analysis. The mechanisms were demonstrated using two applications: a social network derived from the CTS database, and a "star" network derived from the Deeside Insurance data. The analysis demonstrates that the mechanisms may be usefully employed to identify changes in trends discovered in the networks. TM-TFP is able to generate frequent time stamped patterns which can be sub-divided into epochs which may then be compared. By employing the SOM clustering technique, the large number of trend lines that are typically identified may be grouped to facilitate a better understanding of the nature of the trends. Using the proposed cluster comparison/analysis technique, trend migrations can be discovered. The research team is currently developing further methods in which change detection and visualization of the clustering result can be more effective with respect to the requirements of decision makers and stakeholders.

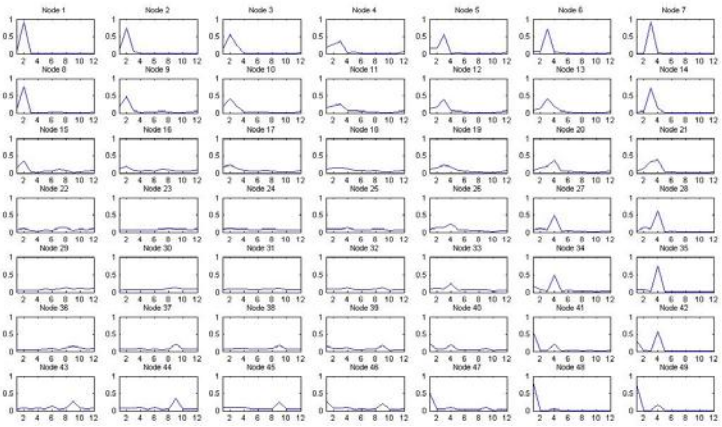


Fig. 6 Deeside Insurance prototype map

References

1. Google Trends. http://www.google.com/intl/en/trends/about.html
2. Streibel, O.: Trend Mining with Semantic-Based Learning. Proceedings of CAiSE-DC (2008)
3. Khan, M.S., Coenen, F., Reid, D., Tawfik, H., Patel, R., Lawson, A.: A Sliding Windows based Dual Support Framework for Discovering Emerging Trends from Temporal Data. Research and Development in Intelligent Systems XXVII, Springer London, pp 35-48 (2010)
4. Raza, J. and Liyanage, J. P.: An integrated qualitative trend analysis approach to identify process abnormalities: a case of oil export pumps in an offshore oil and gas production facility. Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering, Professional Engineering Publishing, vol 223 (4), pp 251-258 (2008)
5. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications. Cambridge University Press (2006)
6. Lauw, H., Lim, E., Pang, H., Tan T.: Social Network Discovery by Mining Spatio-Temporal Events. Computational Mathematical Organization Theory, vol 11(2), pp. 97-118. Springer Netherlands (2005)
7. Agrawal, R., Imielinski, T., and Swami, A. Mining Association Rules between Sets of Items in Large Databases. In Proceedings of ACM SIGMOD Conference (1993)
8. Agrawal, R. andSrikant, R.: Mining sequential patterns. 11th International Conference on Data Engineering (1995)
9. Mannila, H., Toivonen, H., and Verkamo, A.: Discovery of Frequent Episodes in Event Sequences. Data Mining and Knowledge Discovery 1, pp 259289(1997)
10. Dong, G., and Li, J.: Efficient Mining of Emerging Patterns: Discovering Trends and Differences. In Proceeding of fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (1999)
11. Coenen, F.P., Goulbourne, G., Leng, P.: Computing Association Rules Using Partial Totals. Principles of Data Mining and Knowledge Discovery. LNCS, vol. 2168, pp. 54-66. Springer Berlin / Heidelberg (2001)
12. Kohonen, T.: The Self Organizing Maps. Neurocomputing Elsevier Science, vol. 21, pp. 1-6 (1998)
13. Kohonen, T.: The Self Organizing Maps. Series in Information Sciences, vol. 30. Springer, Heidelberg (1995)
14. Wang, J., Delabie, J., Aasheim, H.C., Smel, E., Myklebost, O.: Clustering of the SOM easily reveals distinct gene expression patterns: results of a reanalysis of lymphoma study. BMC Bioinformatics, vol 3(36) (2002)
15. Yan, S., Abidi, S.S.R, Artes, P.H.: Analyzing Sub-Classifications of Glaucoma via SOM Based Clustering of Optic Nerve Images. Studies in Health Technology and Informatics, vol 116 pp 483-488 (2005)
16. Cottrell, M., Rousset, P.: A powerful Tool for Analyzing and Representing Multidimensional Quantitative and Qualitative Data. In Proceedings of IWANN 97. LNCS, vol. 1240, pp. 861-871. Springer Berlin / Heidelberg (1997)
17. Kohonen, T., Oja, E., Simula, O., Visa, A., Kangas, J.: Engineering applications of the Self-Organizing Map. Proceedings of the IEEE, vol. 84(10), pp. 1358-1384 (1996)
18. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications. Cambridge University Press (2006)
19. Lingras, P., Hogo, M. and Snorek, M.: Temporal Cluster Migration Matrices for Web Usage Mining. In Proceedings of IEEE/WIC/ACM InternationalConference on Web Intelligence (2004)
20. Denny, Williams, G.J and Christen, P.: ReDSOM: relative density visualization of temporal changes in cluster structures using self-organizing maps. IEEE International Conference on Data Mining (ICDM), IEEE Computer Society, pp 173-182 (2008)
21. Hido, S., Id T., Kashima, H., Kubo H. and Matsuzawa, H.: Unsupervised changes analysis using supervised learning. Advances in Knowledge Discovery and Data Mining, 12th Pacific-Asia Conference. PAKDD. LNCS, vol. 5012, pp 148-159 (2008)

Retinal Image Classification for the Screening of Age-Related Macular Degeneration

Mohd Hanafi Ahmad Hijazi, Frans Coenen and Yalin Zheng

Abstract Age-related Macular Degeneration (AMD) is the most common cause of blindness in old-age. Early identification of AMD can allow for mitigation (but not cure). One of the fist symptoms of AMD is the presence of fatty deposits, called *drusen*, on the retina. The presence of drusen may be identified through inspection of retina images. Given the aging global population, the prevalence of AMD is increasing. Many health authorities therefore run screening programmes. The automation, or at least partial automation, of retina image screening is therefore seen as beneficial. This paper describes a Case Based Reasoning (CBR) approach to retina image classification to provide support for AMD screening programmes. In the proposed approach images are represented in the form of spatial-histograms that store both colour and spatial image information. Each retina image is represented using a series of histograms each encapsulated as a time series curve. The Case Base (CB) is populated with a labelled set of such curves. New cases are classified by finding the most similar case (curve) in the CB. Similarity checking is achieved using the Dynamic Time warping (DTW).

Mohd Hanafi Ahmad Hijazi
Department of Computer Science, The University of Liverpool, UK e-mail: m.ahmad-hijazi@liverpool.ac.uk

Frans Coenen
Department of Computer Science, The University of Liverpool, UK e-mail: coenen@liverpool.ac.uk

Yalin Zheng
Ophthalmology Research Unit, School of Clinical Sciences, The University of Liverpool, UK e-mail: yzheng@liverpool.ac.uk

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_25,

1 Introduction

Age-related Macular Degeneration (AMD) is the leading cause of blindness in people over 50 years of age. It is caused by damage to the macula, a small area on the human retina that is responsible for seeing fine detail and colour [20]. Although there is no cure for AMD, the condition can be mitigated against in the event of early detection. One of the first symptoms of AMD is the presence of fatty deposits, called *drusen*, on the retina. These can be detected by inspection of retina images routinely collected within screening programmes. This image inspection is conducted manually by trained clinicians. This paper describes an image classification mechanism to (at least partially) automate the identification of drusen in retina images.

The main challenge of the retina image AMD classification problem is that it is often difficult to distinguish drusen from background noise. The need for appropriate image representations, to facilitate the application of data mining, has been identified as a generic challenge within the context of medical image classification [9, 19]. In the context of AMD screening "standard" object segmentation techniques were deemed to be unsuitable as the shape and size of drusen varies significantly from image to image and tends to "blur" into the background. A spatial-histogram [18, 26] based approach was therefore adopted, a technique that features the ability to maintain spatial information between groups of pixels [3]. A *region* based approach is advocated in this paper where by the images are subdivided into "areas" and histograms are generated for each. The histograms were conceptualised as time series where the X-axis represents the histogram "bin" number, and the Y-axis the size of the bins (number of pixels contained in each).

To facilitate the desired classification a Case Based Reasoning (CBR) approach was adopted [21] , where-by a collection of labelled cases were stored in a repository. A new case to be classified (labelled) is compared with the cases contained in this repository and the label associated with the *most similar* case selected. Given that the histograms can be conceptualised as time series, a Dynamic Time Warping (DTW) technique [1, 25] was adopted to determine the similarity between "curves". The principal contributions of the work described are:

- A novel approach to AMD screening.
- A mechanism (that also has wider application) for classifying retina images for AMD without specifically identifying drusen.
- The use of regions in the representation to enhance the classification accuracy.
- An approach to CBR case similarity checking using a time series analysis technique.

The rest of this paper is organised as follows. Section 2 describes the application domain and Section 3 some relevant previous work. The screening process is described in Section 4. Section 5 and 6 provide further detail of how the retinal images are pre-processed and then transformed into the spatial-histogram (time series) representation. The specific classification technique used is described in Section 8, followed by an evaluation of the proposed approach in Section 9. Some conclusions are presented in Section 10.

2 Age-related Macular Degeneration

The work described in this paper is focused on the classification of retinal images, in particular the identification of age-related macular degeneration (AMD). Figure 1 illustrates a typical cross sectional view of the eye. The eye consists of the cornea, iris, pupil, lens, vetreous humour and the retina. As shown in Figure 1, centred at the fovea, the macula is a small area at the centre of the retina. It contains the densest photoreceptors and provides "central vision" and "colour vision". Central vision is essential for humans to see fine detail as required by daily tasks such as reading and writing. Sometimes the delicate cells of the macula become damaged and stop functioning properly. There are various conditions for this to occur amongst which AMD is the leading cause of irreversible

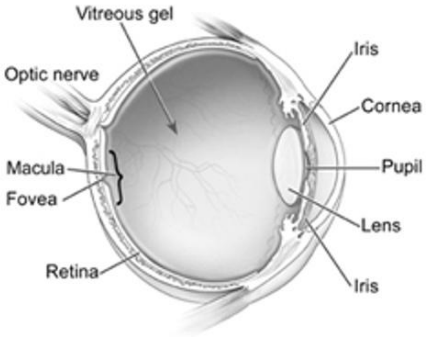
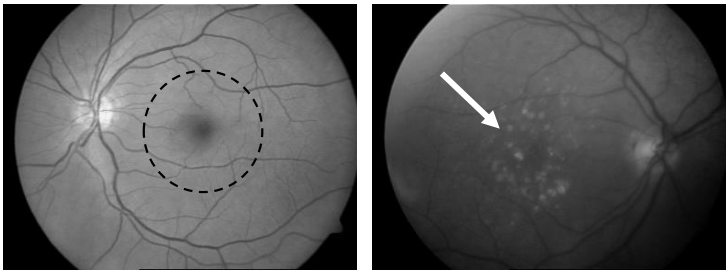


Fig. 1 Cross sectional view of the eye National Institutes of Health (NIH), National Eye Institute (NEI), US (http://www.nei.nih.gov/).

Early diagnosis of AMD is achieved by the identification of *drusen* [20, 8], yellowish-white sub-retinal fatty deposits, by screening patient retinal images. The severity of AMD can be categorised into three classes: *early*, *intermediate*, and *advanced*. AMD can be either *non-neovascular* or *neovascular* [8]. Early AMD is characterised by the existence of several small (63μ m in diameter) or a few medium (63 to 124μ m) sized drusen or retinal pigmentary abnormalities. The presence of at least one large (124μ m) and numerous medium sized drusen, or geographic atrophy, that does not extend to the centre of the macula, characterises intermediate AMD. Advanced non-neovascular (dry) AMD exists once the drusen has reached the center of the macula. *Choroidal neovascularisation* characterizes advanced neovascular (wet) AMD. The drusen itself is often categorised as *hard* and *soft* drusen. Hard drusen have a well defined border, while soft drusen have boundaries that often blend into the retinal background. Figure 2(a) shows an example of normal retinal image with the macula circled. A retina image that features drusen is given in Figure 2(b) (drusen indicated by a white arrow). The classification of AMD images by means of drusen identification is thus not a straightforward process. Most of the previous works have focused on automatic drusen segmentation [4, 13, 22, 23, 29] as opposed to AMD classification. The work proposed here however approaches the AMD screening problem without the need for identification of the physical existence of drusen and aims to classify images as either "AMD" or "non-AMD".



(a) (b)

Fig. 2 Illustration of fundus images in grayscale: (a) Normal and (b) AMD.

3 Previous Work

The earliest work reported in the literature concerning drusen detection is that of Sbeh et al. [30] who used mathematical morphology to identify brightest points to detect drusen. More recent work [4] used a wavelet analysis technique to extract drusen patterns, and multi-level classification (based on various criteria) for drusen categorisation. Other works on the identification of drusen in retina images has focuses on segmentation coupled with image enhancement approaches [22, 23, 29]. Rapantzikos et al. [29] adopted a multilevel histogram equalisation to enhance the image contrast followed by drusen segmentation, in which two types of threshold, global and local, were applied to retinal images. Köse et al. [22, 23] proposed two approaches involving *inverse* drusen segmentation within the macular area. A region growing technique was used to identify "healthy" pixels by applying a threshold on the colour intensity levels [22]. Once this was done, the inverse of the segmented image was used to generate the segmentation of the drusen. A similar inverse segmentation approach, supported by statistical information, was adopted in [23]; where healthy *Characteristic Images* (CIs) were compared to new *Sample Images* (SIs) and a predetermined threshold is applied to classify SI. In [13] another approach, based on a non-parametric technique for anomaly detection, was described that uses a Support Vector Data Description (SVDD) to segment anomalous pixels.

There has been very little reported work on the application of image mining techniques for AMD screening. The existing work (see above) has been mostly focuses on the segmentation/identification of drusen. Of the reported work that the authors' are aware of, only two reports [4, 13] extend drusen detection and segmentation to distinguish retinal images with and without AMD features. However, all the previous work is focused on the detection of drusen using segmentation, a challenging task given the inconsistent visual appearance of drusen and other lesions. The clarity, colour, luminosity and texture of images are affected by several factors during the image acquisition process, such as involuntary eye movement and the media opacity of the subject.

The distinction between the work described here and previous approaches is that we make no attempt to locate and isolate (segment) drusen within retinal images. Instead, we extend the uses of individual colour channel histograms [16] to a spatial-histogram based approach that obviates the need for accurate segmentation

of drusen. Spatial-histograms extend the concept of simple colour histograms by including spatial pixel information [3, 33, 35] and have been shown to perform well in region-based tracking [3], object detection [35] and image retrieval [33].

Space limitations preclude a overview of CBR. However CBR is a well established AI technique with an associated, well established, body of literature. Recommended reference works include [24] and [21]. For a review of the application of CBR in medical domains interested readers are referred to [17] or [2].

4 The AMD Screening Process

An overview of the proposed retinal image classification, to identify AMD, is presented in this section. The approach can be viewed as consisting of two stages, (i) Case Base (CB) generation and (ii) image classification. A block diagram outlining the process is given in Figure 3 (the directed arcs indicate process flow). In the figure the two stages are deliminated by dashed boxes. The case base generation process commences at the top left of the figure, while the classification process at the bottom left.

CB generation comprises three sub-stages: (i) image preprocessing, (ii) histogram generation and (iii) feature selection. CB generation commences with a training set of pre-labelled images which are preprocessed as follows:

1. **Image Enhancement:** Normalisation and enhancement of the image contrast. Colour normalisation is applied first, followed by illumination normalisation and then contrast enhancement to increase the "visibility" of the main retinal anatomy (blood vessels, etc.).
2. **Object Segmentation:** Identification of the main retinal structures.
3. **Noise Reduction:** Removal of blood vessel pixels from the retina images.

The image pre-processing is described in further detail in Section 5.

The next step is to generate the spatial-histograms. In order to make the representation more tractable, colour quantisation was applied to the preprocessed images to reduce the overall dimensionality (number of colours). To generate the histograms the quantised colour retinal images were first partitioned into nine *regions* and then spatial-histograms were extracted for each region. The idea here is that the presence of drusen is often regionalised and consequently we may be more interested in some regions than others. Section 6 gives more detail of the technique used to generate the spatial-histograms.

During feature selection the spatial-histograms (regions) that feature the best discriminatory power (in the context of AMD classification) are identified. The regions are ranked according to their discriminatory power and the top T selected. This process also ensured that the size (number of pixels) of each region/histogram does not bias the resulting classification. The feature selection was conducted using a *class separability* measure which was applied to the collection of histograms representing each retina image and the most appropriate histograms selected. The selected

spatial-histograms were then combined and stored in the form of time series curves (one per image). The feature selection process is discussed in further detail in Section 7.

The image classification task is detailed in Section 8.

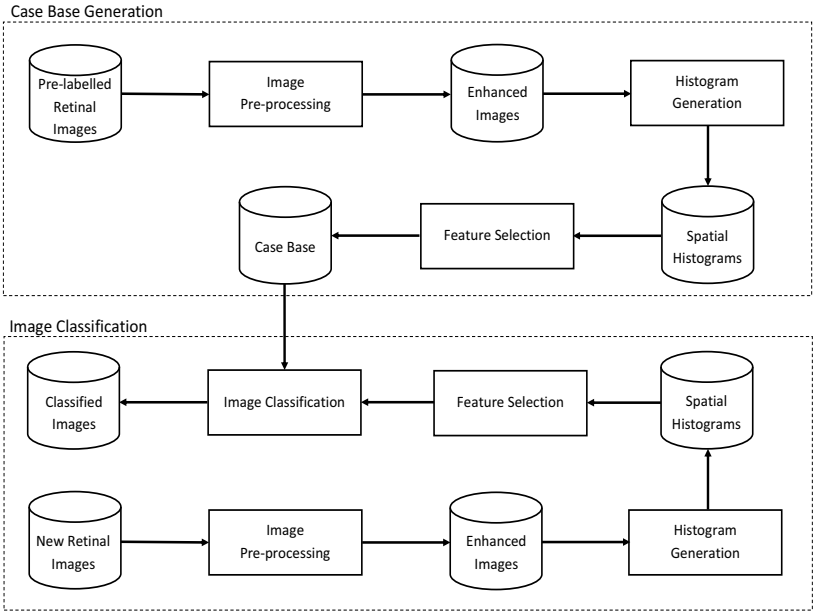


Fig. 3 Block diagram of the proposed retinal images screening system

5 Image Pre-processing

This section describes the image pre-processing steps required to represent images into meaningful forms for image mining. The image pre-processing consists of two steps: (i) image enhancement and (ii) segmentation of anatomic structures to identify retinal blood vessels.

5.1 Image Enhancement

The quality of the retinal images is often severely affected by factors such as: colour variance and non-uniform illumination [11, 27], which are difficult to control. In the context of AMD screening this will lead to difficulties in the detection of drusen, and hamper the associated identification and localisation of retinal common struc-

tures such as retinal blood vessels. Thus, colour and illumination normalisation, and contrast enhancement are important.

Due to the colour variation between different retinal images, colour normalisation must be performed prior to image enhancement. To normalise the colours featured in retinal images a *histogram specification* approach was applied [14]. First, a reference image that represents the best colour distribution and contrast is selected by a trained clinician. Then, the Red-Green-Blue (RGB) colour histograms of the reference image are generated. Finally, the RGB histograms of other images are extracted and each of these histograms is tuned to match the reference image histograms.

Once the colour is normalised, illumination normalisation is applied so as to reduce the luminosity variations on the image. An approach, to estimate the luminosity and contrast variability of the retinal image based on the image background colour, proposed by Foracchia et al. [11] was adopted. This approach estimates the original image, $\bar{I}$, as follows:

$$\bar{I}(x,y) = \frac{I(x,y) - \bar{L}(x,y)}{\bar{C}(x,y)}, \tag{1}$$

where I is the observed image, and $\bar{L}$ and $\bar{C}$ are the estimations of luminosity and contrast, calculated in the neighbourhood N of each pixel. One drawback of this approach is that drusen that are larger than the window size N, used for the estimation, are smoothed in the normalisation process. However, the authors found that this disadvantage could be limited by setting the $\bar{C}$ value to 1 there by excluding the contrast estimation. Contrast normalisation was then conducted using Contrast Limited Adaptive Histogram Equalisation (CLAHE) as described in [36].

5.2 Objects Segmentation

The presence of retinal anatomies, such as blood vessels and the optic disc, sometimes hampers the detection of drusen. The authors' own experiments have indicated that the removal of blood vessel pixels from retina images can improve classification accuracy [16]. This has also been observed more generally by other researchers in the field ([23, 28, 29]).

To segment the retinal blood vessels 2-D Gabor wavelet filters [31] were applied. A pixel is classified as *vessel* or *non-vessel* by means of a Bayesian classifier with a class-conditional probability density function, generated using the Gaussian mixture model. As a result a "retinal vessels" binary representation is generated for each image which is then applied as a "mask" to the enhanced retinal images and consequently the blood vessels pixel values replaced with a "null" value.

The optic disc was however left untouched as experiments conducted by the authors, reported in [16], indicated that removal of the optic disc only results in increased accuracy with respect to a minority of retina images and decreases accuracy with respect to the majority.

6 Spatial Histogram Generation

Colour histograms have been widely used as a simple way of representing images for object identification and retrieval [5, 32]. The main advantage is their robustness against object changes in terms of shape and position within images. The main disadvantage is the loss of spatial information between pixels and colours, thus images with similar histograms may have very different appearances [33, 35]. In some images, the colour distribution of pixels at different sections of an image may be an essential feature that should be included in the image representation. In the context of AMD classification there are a significant number of cases where the AMD images have almost similar colour histograms to the normal ones. The fact that drusen pixel colours are very similar to the colours of pixels adjacent to the retinal blood vessels boundaries (as well the optic disc), may thus lead to classification errors. A spatial-histogram [18, 26] representation was therefore adopted.

The spatial information of an image can be generated by preserving the objects texture and shape using templates [35], as well as by partitioning the image into regions based on the chosen colour values and keep the regions location for each of the chosen colours [18]. The utilisation of texture and shape to extract spatial information is hampered by the nature of the AMD featured images where no common textures and shapes exist, other than the main retinal structures. Therefore, a method to generate colour distribution for each region [33] has been applied in the work described here as it is conjectured that the similar regions of two different classes of retinal images will have different colour distribution. The generation of spatial-histogram consisted of several steps. First, the number of colours was reduced to make the computational cost more feasible. The minimum variance quantisation technique [34], with dithering [10] (implementation using Matlab[1] function *rgb2ind*), was used to reduce the image colours to C colours. A careful selection of C value is essential as it will affect the quality of the generated histograms, as shown in Section 9. The colour quantisation was applied on the global colour space, instead of local, in order to standardise the colour mapping. Thus, all images referenced a similar colour map.

Once the colour quantisation was complete each image was partitioned into N similar sized regions, $R = \{r_1, r_2, ..., r_N\}$, and a spatial-histogram generated for each. The set of spatial histograms for a given image m is defined as:

$$h_m = \{sh_1^m, sh_2^m, ..., sh_N^m\} \tag{2}$$

where sh_n^m is the spatial-histogram generated for region n, $(1 \leq n \leq N)$ in image m with size of C bins. The histogram value for colour c in histogram sh_n^m is then given by:

$$sh_n^m(c) = \alpha \tag{3}$$

[1] http://www.mathworks.com

where α is the c-th bin count in region n of image m, and $(0 \leq c < C)$. The size of each image spatial-histograms, h_m, for an image m is equivalent to $C \times N$; the number of colours, C, multiplied by the number of regions, N. The complete set of histograms representing an image set is then defined as $H = \{h_1, h_2, ..., h_M\}$, where M is the number of images.

7 Feature Selection

Feature selection is a process to reduce the number of features contained in a feature space by removing irrelevant or redundant features[6, 7, 12]. By selecting only those features that have a strong discriminatory power between classes, the computational cost of classification can be considerably reduced while at the same time maximising classification accuracy [6]. Common feature selection techniques[7, 12] include the χ^2 measure, mutual information, Odds Ratio and Principal Component Analysis.

With respect to the AMD screening process described here a class separability method [6] that estimates the affectiveness of a features ability to distinguish between classes using the Kullback-Leibler (KL) distance measure was adopted. This was a two stage process. First an average *signature*, γ_n, histogram was generated for each region with respect to each class as follows:

$$\gamma_n^a = \frac{1}{p} \sum_{j=1}^{p} sh_n^j \tag{4}$$

where n is the region identifier, a is a class label and p is the number of training set images labelled as class a. The *class separability*, $dist_n$, is then calculated by:

$$dist_n = \sum_{a=1}^{d} \sum_{b=1}^{d} \delta_n(a,b) \tag{5}$$

where d is the number of classes and $\delta_n(a,b)$ is the KL distance between histograms of γ_n corresponding to classes a and b described as:

$$\delta_n(a,b) = \sum_{i=1}^{c} p_n(\gamma_n^a(i)) log\left(\frac{p_n(\gamma_n^a(i))}{p_n(\gamma_n^b(i))}\right) \tag{6}$$

where c is the number of bins or colours in the histograms, and $p_n(\gamma_n^a(i))$ is the probability that the n-th feature takes a value in the i-th bin of the signature spatial-histogram γ_n given a class a. The probability, p_n was calculated by dividing each bin count of γ_n by the total number of elements in γ_n.

The features are then sorted in descending order of $dist_n$; the top T features with the highest $dist_n$ provide the best separation between classes and are therefore selected. However, the selection of value of T is domain dependent and for the work described here, $T = 5$ consistently produced the best result as shown in Section 9. The other regions were omitted from further processing. Thus, the size of h_m has

been reduced to only $C \times T$. These histograms then make up the CB for the CBR process.

8 Retinal Image Classification using CBR and DTW

Given a new set of images produced during an AMD screening process these may be classified using the CB developed as described in the foregoing subsections. As noted above the histograms in the CB may be viewed as time series. Similarity checking may therefore be conducted using time series analysis techniques. For the AMD screening a Dynamic Time Warping (DTW) technique [1, 25] was adopted. DTW is a time series analysis technique that measures the distance between two time series through the generation of a *warping path* between these sequences. Given two time series, $T = \{t_1, t_2, ..., t_m\}$ and $\bar{T} = \{\bar{t}_1, \bar{t}_2, ..., \bar{t}_n\}$, a matrix of size $m \times n$ will be formed. The distance between t_i and $\bar{t}_j$, $d(t_i, \bar{t}_j)$, where $0 \leq i < m$ and $0 \leq j < n$ for all i and j is computed using the Euclidean distance similarity measure (other similarity measure methods can also be applied). The minimal warping path is computed by summing up the minimal d for each matrix grid points thus giving a distance between T and $\bar{T}$. More details of the DTW approach with respect to retinal image classification can be found in [15].

9 Evaluation

To evaluate the AMD screening process a collection of 144 retinal images, acquired as part of the ARIA[2] project, were used. The collection was manually pre-labelled, included 86 AMD images and 58 non-AMD images. The experiments described in this section evaluate the performance of the proposed approach. Three metrics are used for evaluation purposes: *Specificity*, *Sensitivity* and *Accuracy*. All experiments were conducted using Tenfold Cross Validation (TCV) whereby the dataset was randomly divided into equal sized "tenths"; and on each TCV iteration, one tenth was used as the test set while the remiander was used as the training set. The objectives of the experiments may be summarised as follows and is described in the following subsections:

1. **Number of Bins Parameter:** To determine the minimum number of bins for the histograms, with respect to colour quantisation, such that classification accuracy would not be adversely affected.
2. **T Parameter Identification:** To determine the most appropriate setting for the T parameter, the threshold that determines the number of regions to be included in the final representation during feature selection.

[2] http://www.eyecharity.com/aria_online

9.1 Number of Bins Parameter

The first set of experiments was designed to determine the number of output bins for colour image quantisation. The aim was to determine the least number of bins while maintaining classification accuracy. Experiments using 32, 64, 128, and 256 bins were conducted (but without the region concept). Table 1 shows the classification results obtained. The results clearly indicate that the overall classification accuracy is relative to the number of bins up to 128. This was expected as low numbers of colour bins will tend to group different coloured pixels in to the same bin, and consequently reduce the discriminative power of the colour representation.

Table 1 Classification results for a range of colour quantisation output bins

Bins	Specificity (%)	Sensitivity (%)	Accuracy (%)
32	53	74	66
64	69	67	68
128	55	81	71
256	52	84	71

9.2 T Parameter Identification

The results presented in the foregoing were generated by setting the number of regions parameter to one. The experiment described in this sub-section consider the effect of using regions, as opposed to the entire image, and how many regions should be considered. For this purpose, the retinal images were partitioned into $3 \times 3 = 9$ regions. The number of regions however could be tailored depending on the problem domain. Spatial-histograms were then generated as described in Section 6. Bin parameter values of 32, 64 and 128 were used; the 256 output bin was omitted from further analysis as it did not give any significant improved performance over the 128 bin threshold and also because it would introduce a significant computational overhead. The retinal image classification was performed using the top-T regions that had the highest discriminatory capability.

Table 2 Classification results for AMD classification using 32 colour output bins with various T values

T	SH-dimension	Specificity (%)	Sensitivity (%)	Accuracy (%)
1	32	71	72	72
2	64	71	65	67
3	96	68	75	72
4	128	68	75	72
5	160	*74*	*79*	*77*
6	192	71	72	72
7	224	69	74	72
8	256	66	74	71
9	288	71	76	74

Table 3 Classification results for AMD classification using 64 colour output bins with various T values

T	SH-dimension	Specificity (%)	Sensitivity (%)	Accuracy (%)
1	64	59	70	65
2	128	66	68	67
3	192	64	71	68
4	256	69	68	69
5	320	*70*	74	73
6	384	69	73	71
7	448	68	70	69
8	512	69	67	68
9	576	69	*76*	*74*

Table 4 Classification results for AMD classification using 128 colour output bins with various T values

T	SH-dimension	Specificity (%)	Sensitivity (%)	Accuracy (%)
1	128	61	69	65
2	256	61	75	69
3	384	59	*81*	72
4	512	65	80	74
5	640	*67*	80	*75*
6	768	64	78	72

Comparisons with various T values are reported in Tables 2, 3 and 4 shows that the average classification results obtained using 32, 64 and 128 sized bins respectively compared to a range of T parameter values. The *SH dimension* column indicates the total number of bins (dimensions) in the spatial-histogram representation (calculated by multiplying the Bin parameter by the T parameter). Inspection of the results indicates that there is a tendency for best results to be produced when $T = 5$, although the evidence is not conclusive. In Table 2 (32 bins) the best results were obtained when $T = 5$, with an overall accuracy of 77%. Similar results are shown in Table 4 (128 bins) with the best overall accuracy of 75% when $T = 5$. The results in Table 3 however performed best with $T = 9$ with overall accuracy of 74%, although a setting of $T = 5$ also produced good results. The best specificity of 74% was recorded with $T = 1$ and 32 colour bins, and the best sensitivity of 81% with $T = 3$ and 128 colour bins. One interesting observation is that specificity tends to increase as the number of colour bins decreases. This may be because a low number of colour bins gives lower colour variation.

Overall the results demonstrate that by using only some portion of the images a comparative or better classification result is generated than when using the entire image. The results in Table 4 contains only six T values (1 to 6) as the machine memory required for the classification process increases quadratically with the size of the colour bins. Thus, the authors have decided to stop the process at $T = 6$ because of: (i) the computational complexity when comparing two spatial-histograms of $T = 6$ with 128 colour output bins is $\bigcirc(n^2)$, the time complexity is more than two orders of magnitude compared to the best results recorded in the experiment ($T = 5$

and 32 colour output bins), and (ii) as indicated by Table 2 and 3 performance will most probably decrease as the size of the spatial-histograms increases.

10 Conclusion

An approach of retinal image classification for AMD screening has been described. The images were represented in the form of spatial-histograms that stored the colour information of the images, while maintaining the spatial information of each colour value. A feature selection strategy, to identify regions in an image that have strong discriminative power to separate classes, was applied to remove irrelevant features, as well as reducing the overall computational cost. The experiments described show both promising and interesting results. Best performance was achieved with a low number of colour bins (32) and a T parameter (number of regions) of 5.

References

1. D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *AAAI Workshop on Knowledge Discovery in Databases*, pages 229–248, 1994.
2. I. I. Bichindaritz and C. C. Marling. Case-based reasoning in the health science: What's next? *Artificial Intelligence in Medicine*, 36(2):127–135, 2006.
3. S. T. Birchfield and S. Rangarajan. Spatial histograms for region-based tracking. *ETRI Journal*, 29(5):697–699, 2007.
4. L. Brandon and A. Hoover. Drusen detection in a retinal image using multi-level analysis. In *Proceedings of Medical Image Computing and Computer-Assisted Intervention*, pages 618–625. Springer-Verlag, 2003.
5. R. Brunelli and O. Mich. Histograms analysis for image retrieval. *Pattern Recognition Letters*, 34:1625–1637, 2001.
6. E. Cantu-Paz. Feature subset selection, class separability, and genetic algorithms. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 959–970, 2004.
7. E. Cantu-Paz, S. Newsam, and C. Kamath. Feature selection in scientific applications. In *Proceedings of 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 788–793, 2004.
8. P. T. V. M. de Jong. Age-related macular degeneration. *The New England Journal of Medicine*, 355(14):1474–1485, 2006.
9. U. M. Fayyad, P. Smyth, N. Weir, and S. Djorgovski. Automated analysis and exploration of image databases: Results, progress, and challenges. *Journal of Intelligent Information Systems*, 4:7–25, 1995.
10. R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial greyscale. *Society for Information Display*, 17(2):75–77, 1976.
11. M. Foracchia, E. Grisan, and A. Ruggeri. Luminosity and contrast normalization in retinal images. *Medical Image Analysis*, 9:179–190, 2005.
12. G. Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Medical Learning Research*, 3:1289–1305, 2003.
13. D. E. Freund, N. Bressler, and P. Burlina. Automated detection of drusen in the macula. In *Proceedings of the Sixth IEEE International Conference on Symposium on Biomedical Imaging: From Nano to Macro*, pages 61–64, 2009.
14. R. C. Gonzalez and R. E. Woods. *Digital image processing*. Pearson Prentice Hall, 2008.

15. M. H. A. Hijazi, F. Coenen, and Y. Zheng. A histogram based approach for the screening of age-related macular degeneration. In *Medical Image Understanding and Analysis 2009*, pages 154–158. BMVA, 2009.
16. M. H. A. Hijazi, F. Coenen, and Y. Zheng. Retinal image classification using a histogram based approach. In *Proc. International Joint Conference on Neural Networks*, pages 3501–3507. IEEE, 2010.
17. A. Holt, I. Bichindaritz, R. Schmidt, and P. Perner. Medical applications in case-based reasoning. *The Knowledge Enginering Review*, 20:289–292, 2005.
18. W. Hsu, S. T Chua, and H. H. Pung. An integrated color-spatial approach to content-based image retrieval. In *Proceedings of the Third International Conference on Multimedia*, pages 305–313, 1995.
19. W. Hsu, M. L. Lee, and J. Zhang. Image mining: Trends and developments. *Intelligent Information Systems*, 19(1):7–23, 2002.
20. R. D. Jager, W. F. Mieler, and J. W. Mieler. Age-related macular degeneration. *The New England Journal of Medicine*, 358(24):2606–2617, 2008.
21. J. Kolodner. *Case-based reasoning*. Morgan Kaufmann, 1993.
22. C. Köse, U. Şevik, and O. Gençalioğlu. Automatic segmentation of age-related macular degeneration in retinal fundus images. *Computers in Biology and Medicine*, 38:611–619, 2008.
23. C. Köse, U. Şevik, and O. Gençalioğlu. A statistical segmentation method for measuring age-related macular degeneration in retinal fundus images. *Journal of Medical Systems*, 34(1):1–13, 2008.
24. D. B. Leake. *Case-based reasoning: Experiences, lessons and future directions*. AAAI Press/MIT Press, 1996.
25. C. S. Myers and L. R. Rabiner. A comparative study of several dynamic time-warping algorithms for connected word recognition. *The Bell System Technical Journal*, 60(7):1389–1409, 1981.
26. B. C. Ooi, K-L. Tan, T. S. Chua, and W. Hsu. Fast image retrieval using color-spatial information. *The International Journal of Very Large Data Bases*, 7(7):115–128, 1998.
27. A. Osareh. *Automated identification of diabetic retinal exudates and the optic disc*. PhD thesis, University of Bristol, UK, 2004.
28. N. Patton, T. M. Aslam, and T. MacGillivray. Retinal image analysis: Concepts, applications and potential. *Progress in Retinal and Eye Research*, 25:99–127, 2006.
29. K. Rapantzikos, M. Zervakis, and K. Balas. Detection and segmentation of drusen deposits on human retina: Potential in the diagnosis of age-related macular degeneration. *Medical Image Analysis*, 7:95–108, 2003.
30. Zakaria Ben Sbeh, Laurent D. Cohen, Gerard Mimoun, and Gabriel Coscas. A new approach of geodesic reconstruction for drusen segmentation in eye fundus images. *IEEE Transactions on Medical Imaging*, 20(12):1321–1333, 2001.
31. J. V. B. Soares, J. J. G. Leandro, R. M. Cesar Jr., H. F. Jelinek, and M. J. Cree. Retinal vessel segmentation using the 2-d gabor wavelet and supervised classification. *IEEE Transactions on Medical Imaging*, 25(9):1214–1222, 2006.
32. M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–31, 1991.
33. H-C. Wu and C-C. Chang. An image retrieval method based on color-complexity and spatial-histogram features. *Fundamenta Informaticae*, 76:481–493, 2007.
34. X. Wu. *Graphic Gems II*, chapter Efficient statistical computations for optimal color quantization, pages 126–133. Elsevier Science and Technology, 1991.
35. H. Zhang, W. Gao, X. Chen, and D. Zhao. Object detection using spatial histograms features. *Image and Vision Computing*, 24:327–341, 2006.
36. K. Zuiderveld. *Contrast limited adaptive histogram equalization*, pages 474–485. Academic Press Graphics Gems Series. Academic Press Professional, Inc., 1994.

An Ensemble Dynamic Time Warping Classifier with Application to Activity Recognition

David McGlynn and Michael G. Madden[1]

Abstract This paper proposes a new ensemble classifier based on Dynamic Time Warping (DTW), and demonstrates how it can be used to combine information from multiple time-series sensors, to relate them to the activities of the person wearing them. The training data for the system comprises a set of short time samples for each sensor and each activity, which are used as templates for DTW, and time series for each sensor are classified by assessing their similarity to these templates. To arrive at a final classification, results from separate classifiers are combined using a voting ensemble. The approach is evaluated on data relating to six different activities of daily living (ADLs) from the MIT Placelab dataset, using hip, thigh and wrist sensors. It is found that the overall average accuracy in recognising all six activities ranges from 45.5% to 57.2% when using individual sensors, but this increases to 84.3% when all three sensors are used together in the ensemble. The results compare well with other published results in which different classification algorithms were used, indicating that the ensemble DTW classification approach is a promising one.

1 Introduction

According to the Kinsella and He [4], there were 560 million people aged 65 years or over in 2008. That figure is expected to double by 2025. This growth will put increasing pressure on the health services of every nation. Elderly people need care and assistance to get about their daily lives [5]. With more people needing care, methods for remote monitoring will be required, along with methods to assess the care needs of a person [3]. Accordingly, there is growing research interest in techniques for remote monitoring of Activities of Daily Living (ADLs), to ensure that such people are able to look after themselves properly. Basic Activities of Daily Living is a term used to describe simple everyday tasks, such as bathroom use, dressing/undressing and washing hands/dishes. Instrumental

[1] National University of Ireland, Galway, University Road, Galway, Ireland
daithi@inbox.com, michael.madden@nuigalway.ie

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_26,

Activities of Daily Living (IADLs) are not necessary for fundamental functioning, but they allow an individual live independently in a community. Examples of these are preparing food and drinks, using the telephone or electronic devices and using the computer. Both Basic and Instrumental Activities of Daily Living are considered in this paper, the six ADLs being: (1) dressing/undressing; (2) bathroom use; (3) washing hands/dishes; (4) preparing food and drink; (5) using computer; and (6) using phone or electronics.

2 Methodology

2.1 Dynamic Time Warping

In this paper we assess the accuracy of Dynamic Time Warping (DTW) in the recognition of ADLs. Dynamic Time Warping is a venerable technique originally developed for speech recognition [6]. It is used to compute the best possible alignment warp between two sequences of data, *T* and *R*, and the associated distortion D(*T, R*). The aim is to find an optimal alignment between variable length sequences $T = \{t_1,..., t_i\}$ and $R = \{r_1,..., r_i\}$. The overall distortion D(*T, R*) is based on a sum of local distances between elements $d(t_i, r_j)$.

When applying this to activity recognition, where we are dealing with 3D data (x, y and z-axis) read from on-body accelerometers, these three values must first be used to calculate the Signal Vector Magnitude (SVM = $\sqrt{x^2 + y^2 + z^2}$). Once this value has been calculated for all the time steps in a sequence *T*, each value is then subtracted from the SVM values in the activity template *R*, to create a matrix. The optimal path through this matrix is then found by identifying the sequence of time steps with the minimum difference between *T* and *R*. The search space was constrained so that only the area close to the diagonal of the matrix was evaluated, thus reducing computation and also avoiding the possibility of false matches arising as a result of any part of the template being allowed match any part of the test vector. Some versions of DTW [7] permit maximum distortion in any direction through the matrix, though this increases computation significantly [8].

2.2 Combining DTW Similarity Scores for Classification

The Single Sensor Classifier algorithm, which is used for the individual sensor tests, is presented below. For each activity, several variables must be initialized: *activity_scores,* which holds the scores of all the stored templates being compared to the test template at each time step; *stored_template* arrays, which hold the

trained templates for each activity; and one *activity_max* variable per activity, for holding the best *activity_score*. Other variables that must be intialised include: *max_score,* for holding the best score out of all the activities at each time step; *max_name,* which will hold the actual name of the best scoring activity; and a *test_template* array the size of the warping window, which will slide through the test dataset.

With all the variables initialised, the classification loop is entered. This loop iterates through the entire *test_dataset*. One element from the *test_dataset* is added to the *test_template* array during each loop. Once the *test_template* array is the size of the warping window, DTW comparisons can take place. The resulting value is then assigned to *activity_score.* Each *activity_score* is compared to the current highest score for that particular activity (*activity_max*). If it is greater, it becomes the new *activity_max*. Once all *activity_score*s for an activity have been evaluated and the *activity_max* established, the *activity_max* is compared to the best score from all activities considered thus far (*max_score*). If *activity_max* is greater than *max_score*, it becomes the new *max_score* and *max_name* is assigned the name of the corresponding activity; e.g. if Bathroom Use was found to be the *max_score,* then *max_name* would be assigned the string "Bathroom Use". When all activities have been considered, the first element in *test_template* is removed prior to a new element being added at the start of the next iteration, as the *test_template* array maintains a sliding window of sensor data. For the current time slice, *max_name* is returned as the best matching activity.

```
Algorithm Single Sensor Classifier

For each Activity
   For x = 0 to number of Stored Templates
      Initialise array stored_template_x[Sliding Window Size] with Training data
      Initialise activity_score[x] = 0
   End For

   Initialise activity_max = 0
End For

Initialise max_name as an empty string
Initialise max_score = 0
Initialise array test_template[Size of Warping Window] to all zero's
Initialise array test_dataset[Size of Test Dataset] with data from Test Dataset

For y = 0 to Size of test_dataset
   Append test_dataset[y] to the end of test_template

   If size of test_template is equal to size of Warping Window
      For each Activity
         For z = 0 to number of Stored Templates
            activity_score[z] ← DTW(stored_template[z], test_template)

            If activity_score[z] > activity_max
               activity_max ← activity_score[z]
            End If
         End For
```

```
        If activity_max > max_score
            max_score ← activity_max
            max_name ← "Activity Name"
        End If
    End For

    Remove test_template[0]
  End If

  Return max_name
End For
```

In the Multiple Sensors Classifier algorithm, we first intialise three *sens_res* arrays. These arrays hold the results from the single sensor tests, as determined in the Single Sensor Classifier algorithm above; i.e. results from the hip, thigh and wrist sensor tests. The *activity_score* variable for each activity must be initialised to 0. These will hold the scores for each activity at each time step in the test. As in the single sensor algorithm above, *max_score* and *max_name* are used to hold the best activity score and the name of the best scoring activity, respectively.

With all variables initialised, the test loop is entered. As all three *sens_res* arrays are of equal size, any of the arrays can be used as a constraint for the number of cycles of the loop. This loop iterates through all three *sens_res* arrays by incrementing y in steps of 6, as there are six activity results to access. Individual activity results are accessed using $y + x$, where x is an offset corresponding to each different activity; e.g. $x = 0$ is *Dressing* result, $x = 1$ is *Bathroom Use* result and so on for the other four activities, up to $x = 5$. The *activity_score* is calculated by averaging the *hip_sens_res*, *thigh_sens_res* and *wrist_sens_res* for each activity at each time step. Each *activity_score* is compared to *max_score* and if greater, it becomes the new *max_score* and *max_name* is assigned the name of the activity. Once all activities have been considered, *max_name* is returned as the best matching activity for that time slice.

```
Algorithm Multiple Sensors Classifier

For each Individual Sensor
  Initialise array sens_res[Number of Results Scores] with results from single sensor tests
End For

For each Activity
  Initialise activity_score = 0
End For

Initialise max_name as an empty string
Initialise max_score = 0

For y = 0 to Size of any sens_res array in increments of 6
  For x = 0 to number of Activities - 1
    activity_score[x] ← average (hip_sens_res[y + x], thigh_sens_res[y + x], wrist_sens_res[y + x])

    If activity_score[x] > max_score
      max_score ← activity_score[x]
```

```
            max_name ← "Activity Name"
        End If

        Increment x by 1
    End For

    Return max_name
End For
```

3 Key Issues

The operation of an Activity Recognition classifier depends on several important issues that are discussed in this section. These include window sizes, similar activities and transitioning between activities.

3.1 Sliding Window & Warping Window Size

The sliding window size is the size of the stored activity template (i.e. number of time steps in a sequence of movements) used to classify the activity being performed by a person. The warping window size is the length of the sequence being dynamically compared to the stored template sequence, if the warping window size is 10%, this will be 10% larger than the sliding window size. A 10% warping window, as used by Sakoe and Chiba [6], is often used, however, Keogh and Ratanamahatana [9] suggest that this is too large and that accuracy peaks at 4%. Tapia [2] found that the optimal window size was 5.6 seconds when classifying posture activities, such as walking, sitting, ascending stairs, using various machine learning classification algorithms. Tapia also found that 22-45 second windows were more suitable for classification of household activities such as gardening, weeding, stacking groceries, bathroom use etc.

While windows of longer length have the advantage of providing more context than smaller ones, they reduce the ability of the system to work in real time. In all the experiments presented in this paper, we use a 10% warping window and a 7 second sliding window.

3.2 Similarities in Activities

An increase in the number of activities naturally increases the difficulty in differentiating some activities, as some may be quite similar to each other. For example, with a short window size of 2 seconds, using a computer mouse could be confused with using a television remote control. Increasing the window size can

help, as no two activities will be composed of identical movements over longer periods. However, a larger window size will increase computation and may be counterproductive, as larger window sizes might contain data from two different activities. Using data from multiple sensors also helps to ameliorate this problem; this is clear from the results presented later in the paper.

3.3 Transitioning between Activities

When a subject is transitioning from one activity to another, it can be difficult to make an accurate classification, as there is a period when data from two activities is being evaluated. Figure 1 shows a graphical representation of a subject transitioning from *Bathroom Use* to *Washing Hands/Dishes*. The x-axis shows *Time* (0.5 second increments) and the y-axis shows the *SVM* value for each time step. It can be seen that the transition between activities occurs around time 24 (12 seconds). With a 7 second sliding window, there will be a period during transition where some data being used to classify the activity will be from *Bathroom Use* and the rest from *Washing Hands/Dishes*. This would result in a high degree of misclassifications during this period. To combat this problem, the classification accuracies are computed for each activity separately, rather than transitioning from one activity to the next.

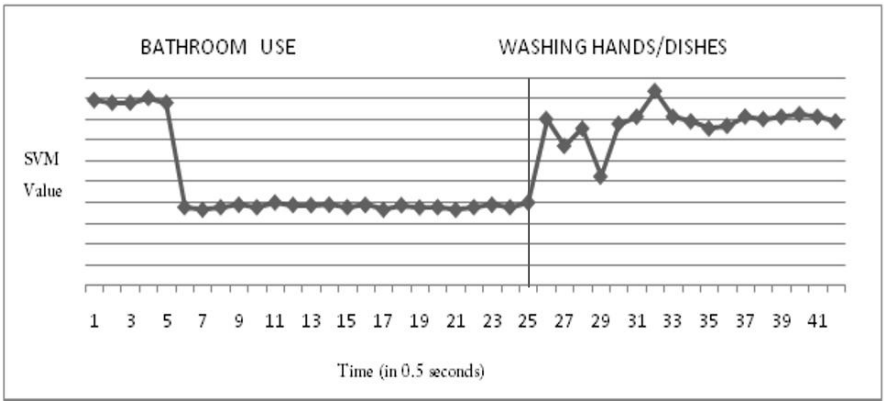


Figure 1 Graph of Transition between Activities

4 Experiments

4.1 Description of Data

The PLCouple1 Placelab dataset [1] was used for this research. This dataset was compiled over a 2.5 month period in 2006 in a live-in laboratory in Cambridge, MA. One month's data has been made publicly available to researchers, 100 hours of which is fully annotated. The data used in this research is from the on-body accelerometer sensors worn by the male participant. Data was read from 3 accelerometers, placed on the participant's Hip, Thigh and Wrist. Wireless sensors transmitted readings to 6 different computers, in real time, averaging 60 ± 20 readings per second. For the work presented in this paper, we down-sampled the data streams to a frequency of 2 readings per second.

4.2 Training Procedure

In the context of DTW, training involves selecting appropriate templates for each activity. This was done by taking a small sample of all the fully annotated data in the dataset and building 10 templates per activity. With the data reduced to 2 readings per second, representative templates of 7 seconds duration (14 time steps) were chosen from random sections of data from different days in the dataset. The data used for training was taken from days that were not used for subsequent testing purposes.

4.3 Testing Procedure

The testing procedure involved testing each activity separately using the longest continuous run of data from the hip, thigh and wrist sensors on the same day. It was important to take the test data for each single sensor test from the same date and time, so that the results could be combined to formulate a classification in the multiple sensor tests. First, individual sensors were tested and the best dynamic time warping score was established for each activity at each time step. Once all individual sensors had been tested, the results were used to calculate an average score from the Hip, Thigh and Wrist sensors in the combined sensor tests. These average scores were then evaluated to get the best score for each time step.

5 Results

Table 1 shows the results of the tests performed using individual sensors and multiple sensors. When testing individual sensors, a 7 second sliding window yielded the following results: Hip 51.36%, Wrist 45.5% and Thigh 57.16%. However, when data from all sensors were combined, an average accuracy of 84.33% for all activities was achieved.

Table 1. Accuracies using 7sec sliding window and subject dependent evaluation

ADLs	HIP	THIGH	WRIST	3xSENSOR
DRESSING	100%	28%	77%	91.86%
WASHING	74%	24%	25%	31.85%
FOOD PREP	16%	99%	29%	99.29%
COMPUTER	9%	95%	43%	98.09%
PHONE	100%	17%	96%	100.00%
BATHROOM	8%	80%	3%	84.91%
AVERAGE	51.36%	57.16%	45.5%	84.33%

5.1 Using Individual Sensors

From the results in Table 1, we see that of the individual sensors, the Thigh sensor produced the most accurate predictions, with an average accuracy of 57.16%, compared to 51.36% for the Hip sensor and 45.5% for the Wrist sensor. Some sensors produced good predictions on certain activities where other sensors performed poorly. This could be due to the similarities in limb movement for various activities; e.g. the wrist movements involved in Washing Hands/Dishes and Food/Drink Preparation are quite similar to each other.

Figures 2, 3 and 4 provide additional details on the individual results. In each case, these figures show a sequence of actual activities at the same points in time. In Figure 3, for example, it can be seen that when the actual activity is *Bathroom Use,* the activity predicted using the hip sensor is most often *Phone/Electronics*. On the other hand, it shows that the hip sensor predicts the *Dressing* activity with high accuracy.

There are some cases in Table 1 where the percentage accuracy is low, even though the actual activity may be a close match to the stored templates. This occurs when several activities are a good match but the predicted activity is not the best match to the actual activity. For example, if the actual activity was

Dressing and the best match returned was *Washing Hands/Dishes* with a 90% similarity score, but *Dressing* was the 2nd best match with an 89.5% similarity score, the final prediction would be incorrect, even though *Dressing* was an excellent match.

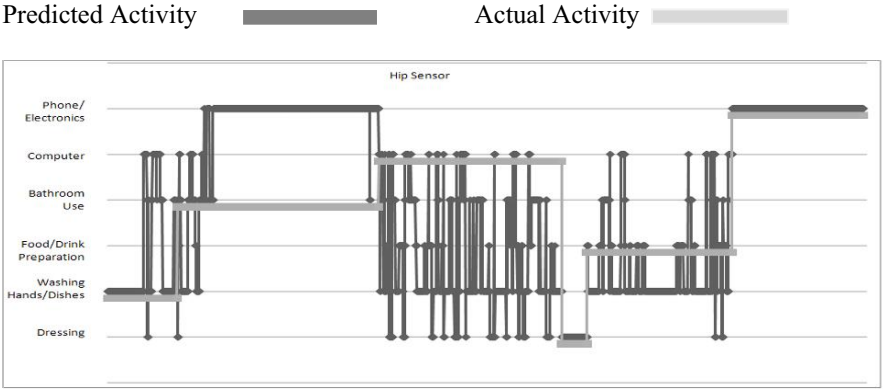


Figure 2 Graphed Results from all tests using only the Hip Sensor

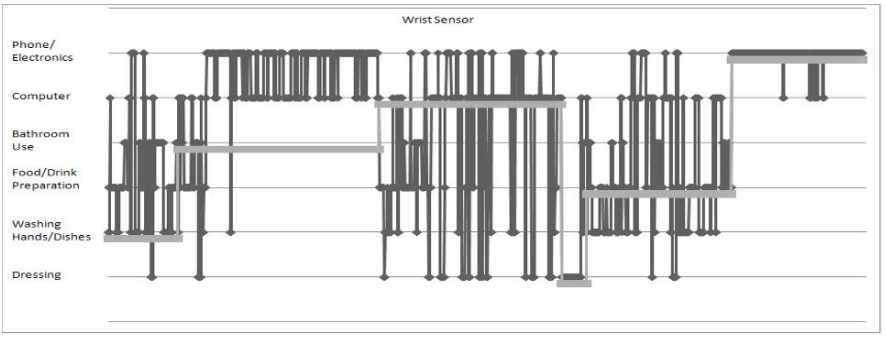


Figure 3 Graphed Results from all tests using only the Wrist Sensor

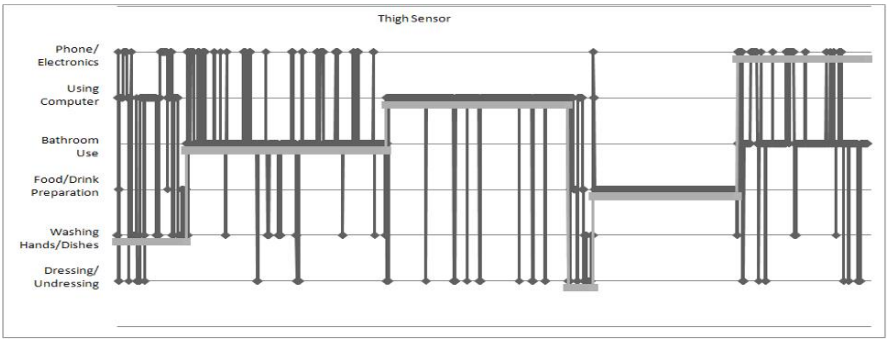


Figure 4 Graphed Results from all tests using only the Thigh Sensor

5.2 Using All Three Sensors

We can see from Table 1 that predicting household activities using only one sensor is not very successful, with accuracies ranging from 45.5% for the wrist sensor to 57.16% for the thigh sensor. However, the table shows that there is a very large improvement in accuracy when all three sensors are used to generate a prediction; the overall average accuracy using all sensors is 84.33%. See Figure 5 below for a visualisation of the results.

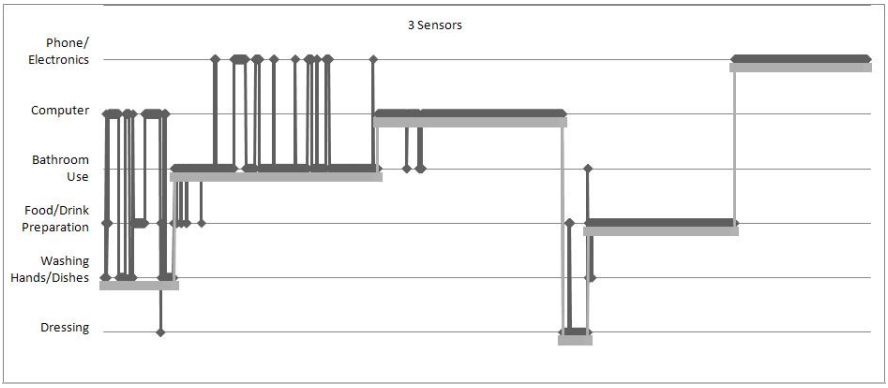


Figure 5 Graphed Results from all tests using 3 sensors

Tables 2-5 below shows the confusion matrices of all the tests performed. Along with Figures 2-5, Tables 2-5 highlight where confusion between activities occur in the classifier system. The results show that using the ensemble classifier with three sensors yields a significant improvement in accuracy of predictions relative to the best sensor in two of the ADLs (*Computer* and *Bathroom*). It also maintains high accuracy in two ADLs (*Food Prep* and *Phone*) that had best individual sensor accuracies of 99-100%. In the case of *Dressing* and *Washing Hands/Dishes,* the ensemble's performance is worse than that of the best individual sensor, but it must be noted that even though the Hip sensor produces the best results for those tasks, it produces very bad results on the *Bathroom* and *Computer* tasks. In the 3xSensor confusion matrix (Table 2), we can see that the *Washing Hands/Dishes* activity's main confusion is with *Computer* and *Food/Drink Preparation*. This could be due to the fact that these three activities consist of similar hand movements. This is also shown in the Wrist Confusion Matrix (Table 4), where we can see that the main confusion is with *Food/Drink Preparation*.

Some individual sensors perform well in classifying specific activities but in general, any one sensor only performs well on two or three of the activities. Accordingly, we conclude that the 3-sensor ensemble classifier is more reliable

overall. It gives consistently good predictions over most activities, relative to any of the three sensors being used individually.

Table 2. Confusion matrix for all activities using 3 sensors

3xSensor Confusion Matrix						
	Dressing	**Washing**	**Food**	**Bathroom**	**Computer**	**Phone**
Dressing	**113**	0	8	2	0	0
Washing	2	**108**	74	0	155	0
Food Prep	0	5	**702**	0	0	0
Bathroom	0	0	13	**833**	0	135
Computer	0	0	0	17	**875**	0
Phone	0	0	0	0	0	**639**

Table 3. Confusion matrix for all activities using Hip sensor

Hi**p Confusion Matrix**						
	Dressing	**Washing**	**Food**	**Bathroom**	**Computer**	**Phone**
Dressing	**123**	0	0	0	0	0
Washing	2	**252**	0	34	51	0
Food Prep	10	462	**118**	61	56	0
Bathroom	3	49	5	**85**	19	820
Computer	106	422	98	180	**86**	0
Phone	0	0	0	0	0	**639**

Table 4. Confusion matrix for all activities using Wrist sensor

Wrist Confusion Matrix						
	Dressing	**Washing**	**Food**	**Bathroom**	**Computer**	**Phone**
Dressing	**95**	5	0	18	5	0
Washing	8	**87**	121	84	19	20
Food Prep	19	210	**211**	62	158	47
Bathroom	13	14	64	**31**	260	599
Computer	108	72	173	33	**388**	118
Phone	0	0	0	0	21	**618**

Table 5. Confusion matrix for all activities using Thigh sensor

Thigh Confusion Matrix						
	Dressing	**Washing**	**Food**	**Bathroom**	**Computer**	**Phone**
Dressing	**35**	22	25	0	41	0
Washing	22	**83**	16	0	156	62
Food Prep	0	1	**700**	0	4	2
Bathroom	18	38	0	**793**	0	132
Computer	35	0	0	1	**856**	0
Phone	32	7	1	488	0	**111**

6 Related Research

A study carried out by Tapia [2] reached similar conclusions to those put forward in this paper, using a completely different classification methodology. When using the C4.5 classifier to predict 51 activities and working with the MIT Energy Expenditure dataset, the following accuracies were reported on predicting household activities: Hip 68.7% ± 11.8%; Wrist 61.5% ± 12.7%; Thigh 59.4% ± 13.3%. Tapia [2] used a 5.6 second window size and the results were averaged over 16 tests, one for each participant in the dataset.

The results reported in this paper using Dynamic Time Warping with individual sensors are slightly lower: Hip 51.36%; Wrist 45.5%; Thigh 57.16%. However, the multiple sensor ensemble DTW classifier proposed in this paper has average accuracy of 84.33%. This is quite similar to the results in Tapia's [2] work when using multiple sensors with different classification algorithms, evaluated on individual subjects: C.4.5 classifier 75.1% ± 9.9%; Naïve Bayes classifier 84.1% ± 7.7%; Logit Boost 82.2% ± 9.6%; Nearest Neighbour 84.0% ± 9.7%.

Boa & Intille [11] have also found that using multiple sensors greatly improved activity recognition performance, on a different dataset. They used 6.7 second windowing and their study comprised of 20 activities performed by 20 subjects, with 5 wireless sensors per subject. In that study, a C4.5 decision tree classifier achieved accuracies of 84.3% ± 5.2% using a leave-one-subject-out evaluation and 71.6% ± 7.4% with user specific training; the latter is more similar to our experimental setup, where the system is calibrated and tested on an individual subject. In the work of Boa & Intille [11], when only Thigh and Wrist sensors were used the accuracies dropped slightly (-3.3% on average) and when individual sensors were used the decrease in accuracy fell dramatically; e.g. Hip (-34.1% compared with all 5); Thigh (–29.5% compared with all 5).

Long et al. [12] conducted a study demonstrating the accuracy achievable using a Naïve Bayes classifier approach compared with a Decision Tree classifier. They

tested on a set of 5 activities performed by 24 subjects. On their dataset, using only one sensor, they report accuracies of 72.8% (Decision Tree) and 71.5% (Naïve Bayes) using a 10-fold cross validation approach.

7 Conclusions

This paper has proposed a new ensemble-based Dynamic Time Warping Classifier, designed for the tasks of recognizing human activities based on wireless wearable sensors. It has shown that this new approach is a useful one, as it yields performance comparable to other state-of-the art methods from the literature that have been applied to similar activity recognition tasks. This paper clearly demonstrates the benefit of using multiple sensors for activity recognition; overall, the multiple-sensor ensemble has an accuracy of 84.33%, compared with the best performing single sensor (Thigh: 57.16%).

The algorithms developed here could be used in the implementation of a fully functional Activity Recognition system, for use in a wide variety of situations, such as remote monitoring of elderly people who are living independently, or monitoring of post-operative patients who are encouraged to walk during their recovery.

References

1. S. S. Intille, K. Larson, E. Munguia Tapia, J. Beaudin, P. Kaushik, J. Nawyn, and R. Rockinson, "Using a live-in laboratory for ubiquitous computing research," in Proceedings of PERVASIVE 2006, Berlin Heidelberg: Springer -Verlag, 2006, pp. 349-365.
2. E.M. Tapia: “Using Machine Learning for Real-time Activity Recognition and Estimation of Energy Expenditure”, Ph.D. Thesis, Massachusetts Institute of Technology, 2008.
3. R. Clark, J.F. Van Nostrand, J.M. Wiener and R.J. Hanley: “Measuring the Activities of Daily Living: Comparisons Across National Surveys”. For U.S. Department of Health and Human Services, 1990.
4. Kevin Kinsella and Wan He, “An Ageing World: 2008. International Population Report”. For the US Census Bureau, June 2009.
5. Laing Buisson Consulting: “Demand for places in elderly care homes projected to increase”. In Care of the Elderly Market Survey, 2006.
6. H. Sakoe & S. Chiba: “Dynamic programming algorithm optimization for spoken word recognition”, IEEE Transactions on Acoustics, Speech, and Signal Processing 26 (1) (1978) 43-49.
7. E.J. Keogh, M.J. Pazzani. “Derivative Dynamic Time Warping”. First SIAM International Conference on Data Mining (SDM'2001), 2001.
8. D. Lemire: “Faster Retrieval with a Two-Pass Dynamic-Time-Warping Lower Bound”, Pattern Recognition, Vol. 42, Issue 9, pp. 2169-2180, Sep. 2009.

9. C.A. Ratanamahatana and E. Keogh: “Three Myths about Dynamic Time Warping Data Mining.” In Proceedings of SIAM International Conference on Data Mining (SDM '05), Newport Beach, CA, April 2005.
10. N. Ravi, N. Dandekar, P. Mysore, and M. Littman, “Activity Recognition from Accelerometer Data”. In Proceedings of American Association of Artificial Intelligence, 2005.
11. L. Bao and S. Intille, “Activity Recognition from User-Annotated Acceleration Data.” In Proceedings of PERVASIVE 2004, Vienna, Austria, April 2004.
12. Xi Long, Bin Yin, and Ronald M. Aarts: “Single-Accelerometer-Based Daily Physical Activity Classification.” In 31st Annual International Conference of the IEEE EMBS, Minneapolis, Minnesota, USA, September 2-6, 2009.

APPLICATIONS OF MACHINE LEARNING II

Self-Adaptive Stepsize Search Applied to Optimal Structural Design

L. Nolle and J. A. Bland[1]

Abstract Structural engineering often involves the design of space frames that are required to resist predefined external forces without exhibiting plastic deformation. The weight of the structure and hence the weight of its constituent members has to be as low as possible for economical reasons without violating any of the load constraints. Design spaces are usually vast and the computational costs for analyzing a single design are usually high. Therefore, not every possible design can be evaluated for real-world problems. In this work, a standard structural design problem, the 25-bar problem, has been solved using self-adaptive stepsize search (SASS), a relatively new search heuristic. This algorithm has only one control parameter and therefore overcomes the drawback of modern search heuristics, i.e. the need to first find a set of optimum control parameter settings for the problem at hand. In this work, SASS outperforms simulated-annealing, genetic algorithms, tabu search and ant colony optimization.

1 Introduction

Structural engineering often involves the design of space frames, or trusses, that have to resist predefined external forces without exhibiting plastic deformation. A truss consists of triangular groups of beams, or members, which are connected at joints, known as nodes, at their ends. External forces are considered to act only on the nodes and result in forces, either compressive or tensile, in the members. All joints are assumed to be revolutes and hence no torsional forces (moments) occur. Figure 1 shows a pylon, a typical example of a truss. The weight of the structure and hence the weight of its constituent members has to be as low as possible for economical reasons without violating any of the load constraints. Usually, the material is specified by the customer or determined by the application. The lengths of the members are also defined by the geometry of the structure and hence the design variables are the cross-section areas of the individual members.

1 Nottingham Trent University, NG11 8NS
{lars.nolle|john.bland}@ntu.ac.uk

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_27,

Figure 1. Pylon, a typical example of a truss

Engineers need to find the minimum cross-section area for each member that does not violate any of the given constraints. The next section introduces a well-known discrete optimization problem, the 25-bar problem.

2 The 25-bar Problem

The 25-bar problem was originally defined by NASA in the 1970s [1]. In this problem, the optimal cross-sections of the members of a three dimensional structure (Figure 2) have to be found so that the total weigh of the structure is minimized.

The structure consists of 25 members of defined length and predefined material properties. The structure has to withstand defined external forces without exhibiting any plastic deformation and the stresses within the members and the displacements of joints have to stay within allowed ranges. The design variables are the cross-sectional areas of the truss members. They can only take values from a finite set of allowed values. This set contains 41 different areas.

For this study, the topology, material properties, loading and constraints are taken from Schmit and Miura [1]. The constant H=0.635m, the material properties and loading are given in Table 1 and Table 2. Due to symmetry in the structure, each member belongs to one of eight groups with the same cross-sectional area. The group members and their comprising member numbers are provided in Table 3.

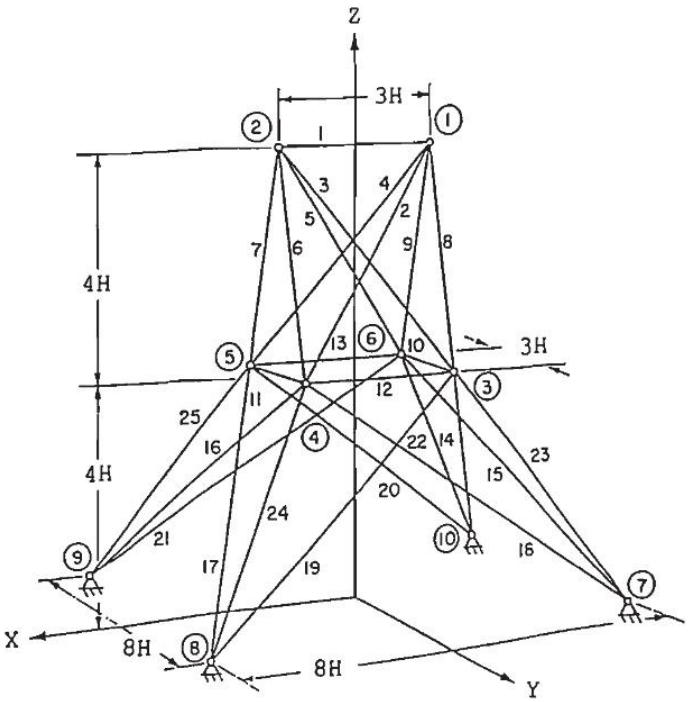


Figure 2. Truss with 25 members (from [5])

Table 1. Material Properties

Material	Aluminum
Young's modulus	E=6.895 x 10^7 kN/m^2
Density	ρ =2767.042 kg/m^3

Table 2. Load components

Joint number	Load component (kN)		
	x	y	z
1	4.4482	44.482	-22.241
2	0.0	44.482	-22.241
3	2.2241	0.0	0.0
6	2.2241	0.0	0.0

Table 3. Member groupings

Group number	Group members
1	1
2	2,3,4,5
3	6,7,8,9
4	10,11
5	12,13
6	14,15,16,17
7	18,19,20,21
8	22,23,24,25

The goal of the optimization is to minimize the total weight m_{total} of the structure (Equation 1).

$$m_{total} = \sum_{i=1}^{25} \rho l_i a_i \tag{1}$$

Where:

ρ : density of member material

l_i : length of *i*th member

a_i : cross-section area of *i*th member

The displacement δ_i for joint *i* has to satisfy the following constraints (Equations 2 and 3).

$$\left|\delta_i\right| = 0 \text{ for joints 7,8,9 and 10} \tag{2}$$

$$\left|\delta_i\right| \le \delta_{\max} \text{ for all other joints} \tag{3}$$

The numerical value for the maximum displacement $\delta_{\max}$ is 0.889 cm.

The stress σ_i for member *i* has to satisfy the following constraints (Equation 4 and Equation 5):

$$\sigma_i \le \sigma_{\max} \text{ if in tension} \tag{4}$$

$$\left|\sigma_i\right| \le \sigma_{cf} \text{ if in compression} \tag{5}$$

In the experiments described below, the numerical value for the maximum stress $\sigma_{\max}$ for members in tension is 2.7579 x 10^5 kN/m^2 whereas for members in

compression the maximum stress σ_{cr} can be calculated using Euler's buckling formula for thin-walled tubular members with a mean radius of R=5.08 cm (Equation 6):

$$\sigma_{cr} = \frac{\pi^2 E R^2}{2 l_i^2} \tag{6}$$

However, for members of groups 6 and 7 Equation 7 was used to determine the stress constraints as it was done originally by Schmit and Miura [1]:

$$|\sigma_i| \leq \begin{cases} 4.6602 \times 10^4 \, kN/m^2 & for\, i = 14 \ldots 17 \\ 4.7781 \times 10^2 \, kN/m^2 & for\, i = 18 \ldots 21 \end{cases} \tag{7}$$

The set of possible areas comprises of 41 discrete values a_n [2] and can be calculated using Equation 8:

$$a_n = 0.064516, 0.64516 \cdot n \quad n = 1,2,\ldots,40 \tag{8}$$

Since there are eight groups and each of them can take one of 41 different cross-section areas, the number of possible designs is $41^8 = 7.98 \text{x} 10^{12}$. This large number prohibits an exhaustive search of the design space for the best design.

Also, the fitness landscape is not convex, i.e. there exist many local optima, which makes it difficult for local optimization techniques.

In the past a number of search meta-heuristics have been successfully used to find near optimum solutions to the 25-bar problem, for example ant colony optimization [2], genetic algorithms [3], simulated annealing [4] and tabu-search [5].

However, all the algorithms have a number of control parameters, which have to be pre-determined before the optimization can take place. Finding suitable parameter settings requires carrying out a large number of experiments. If unsuitable parameter settings are used, it is quite likely that the algorithm fails to find a near-optimal solution.

To overcome this problem, self-adaptive stepsize search has been developed [6,7]. This algorithm has only one control parameter and hence it is relatively easy to adapt to a given problem, i.e. to chose the one control parameter. The next sections explains the algorithm in more detail.

3 Self-Adaptive Stepsize Search

For heuristic search algorithms, like Hill-Climbing (HC), it was previously shown that the definition of the neighbourhood, and in particular the chosen step size, is crucial to the success of the algorithm [6], not only for continuous parameter search, but also for discrete parameters, when the search space is too large to consider direct neighbours of a candidate solution for performance reasons.

It was shown that selection schemes with random step sizes with an upper limit (maximum step size s_{max}) outperform neighbourhood selection schemes with a constant step length. It was also demonstrated that using a scaling function for reducing s_{max} over time could again increase the performance of Hill-Climbing algorithms.

However, it would clearly be of benefit if the maximum step length would be more adaptive to the search progress itself. Therefore, a new population-based adaptation scheme with a self-adaptive step size, referred to as Self-Adaptive Step-size Search (SASS) has been developed for HC [7], where the temporary neighbourhood of a particle p_i is determined by the distance between itself and a randomly selected sample particle s_i of the population during each iteration.

At the beginning of a search this distance is likely to be large, because the initial population is uniformly distributed over the search space and the chances are high that s_i is drawn from a different region within the input space.

When the search is progressing, each particle is attracted by a local optimum and hence the population is clustered around a number of optima. If both, p_i and s_i are located in different clusters, p_i has the chance to escape its local optimum if it samples from a region with a higher fitness, i.e. lower costs.

Towards the end of the search, most particles have reached the region of the global optimum and hence their mean distance is much smaller than in the initial population.

As a result, the maximum step size s_{max} is sufficiently small to yield the global optimum. Figure 3 shows pseudo code of the algorithm. The main advantage of SASS is, that it only has one control parameter that has to be chosen in advance, which is the number of particles n in the population.

The processing time is proportional to the number of particles and hence the complexity of the algorithm is $O(n)$.

```
Procedure selfAdaptiveStepSizeSearch
Begin
  initialise population of n particles
  While stopping criterion not met
  Begin
    For every particle p in population
    Begin
      select random particle s ≠ p
      For every component p_i in particle p
      Begin
        s_max ← | p_i − s_i |
        generate random value r ∈ [-s_max; +s_max]
        p'_i ← p_i + r
      End
      If f(p') better than f(p) then p ← p'
    End
  End
  Return best result
End
```

Figure 3. Pseudo code for SASS algorithm

Although the algorithm is very simple, it has been shown that it is capable of finding near optimal solutions without the need of experimenting with different control parameter combinations [7]. In the next section, SASS is applied to the 25-bar problem and the results are compared with the results reported in the literature.

4 Experimental Results

In order to prove that SASS does not need computational expensive pre-determining of control parameters for practical applications, the number of particles was chosen to be 100 in the experiments, based on previous experience, i.e. the number was chosen to be approximately ten times the order of magnitude of the size of the design or input space.

The experiment was repeated 40 times and the results are presented in Table 4. The best solutions, i.e. the solutions with the lowest weights, are highlighted.

Table 4. Optimization results

Experiment #	Weight in kg	Experiment #	Weight in kg	Experiment #	Weight in kg	Experiment #	Weight in kg
1	**208.42**	11	**208.42**	21	**208.42**	31	**208.42**
2	**208.42**	12	208.62	22	208.92	32	**208.42**
3	**208.42**	13	208.62	23	**208.42**	33	208.62
4	208.62	14	208.62	24	209.31	34	**208.42**
5	208.82	15	208.62	25	208.62	35	208.82
6	208.53	16	208.53	26	208.62	36	**208.42**
7	208.62	17	**208.42**	27	**208.42**	37	208.92
8	**208.42**	18	**208.42**	28	208.62	38	**208.42**
9	**208.42**	19	**208.42**	29	**208.42**	39	208.62
10	208.82	20	**208.42**	30	208.72	40	**208.42**

As it can be seen from Table 4, SASS was able to find the minimum solution of 208.42 kg in 50% of the experiments and came close to the optimum solution in all the other experiments. Table 5 summarizes the results obtained:

Table 5. Summary of results

Best [kg]	Mean [kg]	Standard Deviation
208.42	208.57	0.193

The mean value of the weights achieved is lower than the best solution reported in the literature (Table 6). Also, the standard deviation achieved is 0.193, i.e. in most of the experiments the algorithm found a near optimum solution. Only in experiment 24 the solution was notably worse but still better than any of the reported solutions from the literature.

Table 6. Comparison of algorithms

Algorithm	Reference	Best [kg]
ACCESS1	Schmit and Miura [1]	247.3
Simulated Annealing	Bennage and Dhingra [4]	218.3
Genetic Algorithm	Rajeev and Krishnamoorthy [3]	247.7
Tabu Search	Bland [5]	211.3
Ant Colony Optimisation	Bland [2]	209.3
SASS	present study	208.4

Table 7 shows a comparison of the best design found by Ant Colony Optimization and the best design found by SASS. As it can be seen from the table, both designs are very similar. Only groups 3 and 5 differ slightly, which indicates that ACO has reached the area of the search space that contains the global optima, but failed to exploit this region.

Table 7. Comparison of best designs found by Ant Colony Optimization and SASS

Group	Cross-section area in cm^2	
	ACO	SASS
1	0.064516	0.064516
2	0.64516	0.64516
3	**21.93544**	**23.2258**
4	0.064516	0.064516
5	**16.12900**	**11.6129**
6	5.16128	5.16128
7	0.64516	0.64516
8	25.16124	25.16124

5 Conclusions

In this work the well-known 25-bar problem, which was originally devised by NASA, has been solved using self-adaptive stepsize search (SASS), a relatively new population-based variant of the hill-climbing algorithm. This algorithm overcomes the drawback of hill-climbing, which is that it converges toward the next local optimum, by dynamically adapting the stepsize used during the search. That enables SASS to jump over local optima whilst still being able to home in on the global optimum.

A common drawback of modern optimization heuristics is that these algorithms usually require a large number of experiments in order to find a suitable control parameter set. This can be seen as an optimization problem itself and hence requires a considerable amount of time and afford. This is a real obstacle for practitioners who simply want to use heuristic search as a problem-solving black-box tool.

SASS on the other hand, has only one control parameter, the population size. However, there is a trade off between effectiveness and efficiency; if the population size is small, the algorithm is more likely to converge towards a local optimum whereas a large population size leads to a large number of design evaluations.

In order to prove that SASS is robust and suitable for use in practical applications by non-experts, the population size was not determined by experiments in this work but was chosen rather arbitrarily based on previous experience. It was demonstrated that SASS outperforms simulated-annealing, genetic algorithms, tabu search and ant colony optimization for the 25-bar problem without the need of fine-tuning the control parameter.

Therefore, it was shown that SASS can be used by inexperienced practitioners as an effective and efficient black-box optimizer, without the need to become an expert in computational optimization.

References

1. Schmit, L.A., Miura, H.: Approximation Concepts for Efficient Structural Syntheiss, NASA Contractor Report CR-2552 (1976).
2. Bland, J.A.: Optimal structural design by ant colony optimization, Engineering Optimization, Vol 33, p 425-443 (2001).
3. Rajeev, S., Krishanomoorthy, C.S.: Discrete optimization of structures using genetic algorithms, J Struct Eng, ASCE, Vol 118, pp 1233-1250 (1992).
4. Bennage, W.A., Dhingra, A.K.: Single and multiobjective structural optimization indiscrete-continuoius variables using simulated annealing, Int J Num Math Eng, Vol 38, pp 2753-2773 (1995).
5. Bland, J.A.: Discrete-variable optimal structural design using tabu search, Structural Optimisation, Vol 10, pp 87-93 (1995).
6. Nolle, L.: On the Effect of Step Width Selection Schemes on the Performance of Stochastic Local Search Strategies, Proceedings of the 18th European Simulation Multiconference ESM 2004, Magdeburg, Germany, pp 149-153 (2004).
7. Nolle, L.: On a Hill-Climbing Algorithm with Adaptive Step Size: Towards a Control Parameter-Less Black-Box Optimisation Algorithm, Reusch, B. (Ed): Computational Intelligence, Theory and Applications, Advances in Soft Computing, Vol. 38 (2006).

Health Problems Discovery from Motion-Capture Data of Elderly

B. Pogorelc[1] **and M. Gams**[2]

Abstract Rapid aging of the population of the developed countries could exceed the society's capacity for taking care for them. In order to help solving this problem, we propose a system for automatic discovery of health problems from motion-capture data of gait of elderly. The gait of the user is captured with the motion capture system, which consists of tags attached to the body and sensors situated in the apartment. Position of the tags is acquired by the sensors and the resulting time series of position coordinates are analyzed with machine learning algorithms in order to identify the specific health problem. We propose novel features for training a machine learning classifier that classifies the user's gait into: i) normal, ii) with hemiplegia, iii) with Parkinson's disease, iv) with pain in the back and v) with pain in the leg. Results show that naive Bayes needs more tags and less noise to reach classification accuracy of 98 % than support vector machines for 99 %.

1 Introduction

Increasing of life expectancy and decreasing of birth rate are main causes for aging of population of the developed countries. Consequently, working-age population's capacity for taking care of its elderly members is decreasing [1].

Moreover, elderly want to live at their homes for as long as possible. When living alone, nobody could detect the potential health problem of the elderly. To overcome this problem, we propose system for automatic ubiquitous elderly health care, for which we developed accurate techniques for recognition of common health problems manifesting in gait of elderly. In case the system would recognize

1 Jožef Stefan Institute, Department of Intelligent systems, Ljubljana, Slovenia & Špica International, Ljubljana, Slovenia
bogdan.pogorelc@ijs.si

2 Jožef Stefan Institute, Department of Intelligent systems, Ljubljana, Slovenia & Špica International, Ljubljana, Slovenia
matjaz.gams@ijs.si

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_28,

the health problem, it would automatically notify the physician and show him/her the explanation of the automatic diagnose in form of visualization of the kinematic model. Therefore, elderly would get constant health care at their homes and physicians would be less (over)occupied with work, however they would still have the possibility to confirm/reject automatic diagnose. In this case elderly would gain constant (ubiquitous) support, providing them more safety and confidence in living at their homes.

The target health problems for automatic recognition are: hemiplegia (usually the result of stroke), Parkinson's disease, pain in the leg and pain in the back. The gait of the user is captured with the motion capture system, which consists of the tags attached to the body. The position of the tags is acquired by the sensors situated in the apartment and the resulting time series of position coordinates are analyzed with machine learning algorithms in order to recognize the specific health problem.

First objective of the research is to discover to what extent the automatic recognition of health problems with motion capture system is feasible. Second objective is to investigate the classification accuracy achievable using various numbers/placements of tags on the user's body and various amounts of noise in tag coordinates. Tag placement must achieve a trade-off between usability and technical requirements – the users prefer as few tags as possible, but too few tags cannot ensure sufficient accuracy. Both the finding regarding noise and tag placement can affect motion-capture system selection and further development and applications of care systems for elderly.

The paper is organized as follows. In Section 2 we present related work from the fields of movement recognition and sensor appliances used for the task of human behavior movement recognition. In Section 3 we propose our movement recognition system, which is based on the novel features which we describe in that section. We also define the machine learning setting, which we use for prediction of previously chosen health problems. In Section 4 we evaluate the performance of machine learning classifiers. In Section 5 we propose prototype application for the explanation of the interpreted health state. Section 6 concludes our paper and presents the potential further work in this field.

2 Related Work

Motion capture. For the automatic recognition of the movement (gait) pattern, the movement must first be captured. For this purpose many types of motion capture devices exist. Widely used are inertial sensors composed of accelerometers or gyro sensors [2]-[4]. The second widely used approach uses video-image processing for the reconstruction of the human body movement [22], [23]. The third approach uses cameras in combination with tags attached to the

body. Usually infra-red (IR) cameras are used and the body posture is reconstructed from the position of the retroreflective tags [12], as in our approach.

There also exist some specific measurement devices for the recognition of tremor – a symptom in Parkinson's disease, but not in hemiplegia, pain in the leg and pain in the back. Tremor can be acquired with variety of measuring approaches, including sensors for a measurement of the angle of the joint deflection in tremor-type joint movements [5] and with electromyography [6].

We did not address the recognition of the activities of daily living, such as walking, sitting, lying, etc., and the detection of falling, which has already been solved [9], [15]. We were focused on solving a more challenging task, which is the recognition of
gait-related health problems.

Recognition of health problems. In related work [10], [11], physicians and physical therapists usually diagnose health problems which manifest in gait just by manually observing the user's gait. If they cannot decide for certain diagnose easily, they can use semiautomatic approach. In this approach, they capture the movement using motion capture system and then they analyze it by manual observing the time series of the movement (such as time series of angles of legs, etc.).

They use the same procedure for pre-rehabilitation planning (e.g. physical therapy) and after- rehabilitation evaluation to calculate the difference in movement between those states. They also use some special medical tests, e.g., test for assessment of balance. However, none of those approaches can provide constant real-time observation of the elderly at home, for fast recognition of changes in movement (gait), indicating some health problem or progress in at home rehabilitation.

The example of the semiautomatic approach is also [23], where a system for long-term monitoring of the gait in Parkinson's disease is presented. The characteristics of every stride taken were acquired using a lightweight ankle-mounted sensor array that saved the data into a small pocket PC. In comparison to our approach, the described approach was meant for the monitoring of progress of Parkinson's disease for the known Parkinson's disease patient and was not used for the early automatic recognition of health problems.

The work [5] presents sensors for the measurement of the angle of joint deflection in tremor-type joint movements, which can also be used to assess Parkinson's disease. However, the sensor systems are too big and would prevent users from doing the activities of daily living if the systems were to be worn all day. Just like the system described previously it has major drawback in comparison to our approach, because the system cannot automatically recognize Parkinson's disease or any other health problem.

Using similar motion-capture system as that in our approach, the automatic distinguishing between health problems such as hemiplegia and diplegia is presented in [14]. A classification accuracy of 92.5 % was reported. This was

achieved with Self-Organizing Maps, whose features were wavelet-transformed gait characteristics such as walking speed and stride length.

To present the motion recognition field more generally, also fall detection and activity recognition will be mentioned here. The works are broken down by the choice of motion capture system.

Fall detection and activity recognition using inertial sensors. Fall detection with accelerometers and gyro sensors [20] is quite common, particularly when using simple threshold algorithms [13], achieving classification accuracies close to 100 %. With a more advanced approach using the One-Class SVM machine learning algorithm whose features were accelerations, changes in acceleration etc., an accuracy of 96.7 % was reported [22].

Accelerometers can also be used for activity recognition. Five tri-axial accelerometers distinguished 30 physical activities of various intensities with an accuracy of 94.9 % with person-dependent training and 56.3 % with person-independent training [19]. This was achieved with C4.5 decision trees using various time- and frequency-domain features.

Fall detection and activity recognition from video. Fall detection and activity recognition from video are also quite common. In an example of fall detection [20], objects in the video were first identified and then rules were applied to the aspect ratios of their bounding boxes, their horizontal and vertical gradients and angles, achieving an average accuracy of 83 %. In an example of activity recognition [23], ten states related to the activities of daily living were distinguished with an average accuracy of 74.1 %. This was achieved by first reconstructing the human body in the video and then matching its state against models defined using a specialized language.

Fall detection and activity recognition from video and tags. This work, using similar motion capture devices as ours [18], used 43 body tags sampled with 30 Hz to distinguish between seven activities related to military operations, reporting an accuracy of 76.9 %. This was achieved with the SVM machine learning algorithm whose features were the tag coordinates belonging to two postures separated by 1/3 of a second, reduced in number to 20 using Principal Component Analysis.

Another paper on activity recognition [17] used 41 body tags sampled with 120 Hz to distinguish between 21 dance gestures, reporting an accuracy of 99.3 %. The gestures were represented with Gaussian mixture models of joint angles. The high accuracy can in part be attributed to the high quality of the input data, the strictly defined gestures and the fact that testing was always done on the same dancer as the training.

Investigation of the optimal placement of motion capture devices. An important part of the research presented in this paper is the study of the impact of the placement of tags on the user's body and the amount of noise in tag coordinates on the classification accuracy. The closest work in this respect that we are aware of investigated the placement of accelerometers for fall detection [8],

[13]. Their finding was that the head provides optimal accuracy, but is impractical, the wrist is not appropriate, and the waist is a good option.

3 Methods and Materials

3.1 Targeted Health Problems

For the development of our health problem recognition system we focused on four health problems and normal walking as a reference in accordance with the suggestions received from the collaborating medical expert. The specific health problems for recognition were suggested based on the incidence in the elderly aged 65+, the medical significance and the feasibility of their recognition from the observed subjects' movements. The following four health problems were chosen as the most appropriate [10], [11]:

- Parkinson's disease,
- hemiplegia,
- pain in the leg and
- pain in the back.

3.2 Construction of the Features for Machine Learning

A physician usually diagnoses target health problems while observing a patient's gait (i.e. posture and the walking pattern). Since the gaits of patients with the observed four health problems and normal gait look similar to each other, a physician needs to pay attention to many details that need to be transformed into computable quantities. In practice, the observed five health states can be detected by the distinctive walking patterns 9, 10. For the task of the automatic health-problem recognition we proposed and tested 13 features that are based on the tag locations, for 12 tags, placed on the shoulders, elbows, wrists, hips, knees and ankles of the elderly.

The proposed features are listed as follows:

- Absolute difference between i) average distance between right elbow and right hip and ii) average distance between right wrist and left hip.
- Average angle of the right elbow.
- Quotient between maximal angle of the left knee and maximal angle of the right knee.
- Difference between maximal and minimal angle of the right knee.

- Difference between maximal and minimal height of the left shoulder.
- Difference between maximal and minimal height of the right shoulder.
- Quotient between i) difference between maximal and minimal height of left ankle and ii) maximal and minimal height of right ankle.
- Absolute difference between i) difference between maximal and minimal speed (magnitude of velocity) of the left ankle and ii) difference between maximal and minimal speed of the right ankle.
- Absolute difference between i) average distance between right shoulder and right elbow and ii) average distance between left shoulder and right wrist.
- Average speed (magnitude of velocity) of the right wrist.
- Frequency of angle of the right elbow passing average angle of the right elbow.
- Average angle between i) vector between right shoulder and right hip and ii) vector between right shoulder and right wrist.
- Difference between average height of the right shoulder and average height of the left shoulder.

The features for the identification of the chosen four health problems were designed with the help of a medical expert. They are afterwards used for modeling using the machine learning methods.

3.3 Modeling Target Health Problems using Machine Learning

To construct a predictive model which can be subsequently used to automatically recognize health problems in the subjects yet to be observed (our first objective from the Introduction), we employed supervised learning methods from the field of machine learning (sub-field of the artificial intelligence field). In supervised learning, a training data set of already labeled subjects (i.e. classified into one of the target five classes) is used to construct a model, which is later used to predict the class of the subjects for which we wish to detect the health problem.

Our task was therefore to classify the recordings of walking into five classes: four with selected health problems (classes *hemiplegia, parkinson, pain-leg, pain-back*) and one without it (*normal*).

Data for the evaluation of the proposed approach was collected by recording the gaits of test subjects with particular walking patterns, of which each subject was recorded 4-5 times. The final data set of 141 recordings consisted of:

- 25 recordings of normal walking,
- 45 recordings of walking with hemiplegia,
- 25 recordings of walking with Parkinson's disease,
- 25 recordings of walking with a limp due to a pain in the leg,
- 21 recordings of walking with a limp due to a pain in the back.

The recordings consisted of the position coordinates for the 12 tags worn on the body, sampled with 60 Hz. The tag coordinates were acquired with Smart IR motion capture system with 0.5 mm standard deviation of noise.

For each subject, the locations of the sensor tags were recorded in a session which lasted 5-8 seconds from which a vector of 13 proposed features was computed. These learning examples were labeled with the type of the representing health problem, yielding the final data on which the classifier was trained.

4 Experiments and results

In our experimental work we focused on analyzing the classification accuracies of model, built using the machine learning methods. The experimental classification accuracies were obtained using stratified 10-fold cross validation. We compared naive Bayes and support vector machines machine learning algorithms 20.

The 10-fold cross-validation resulted in classification accuracy of 97.2 % and 97.9 % for naive Bayes and support vector machines classifier, respectively.

Table 1. Confusion matrices of matrices of naive Bayes (left) and support vector machines classifier (right), where H=hemiplegia, L=pain in the leg, N=normal (healthy) walking, P=Parkinson's disease and B=Pain in the back. Numbers denote quantity of the classified examples.

		classified as				
		H	**L**	**N**	**P**	**B**
true class	**H**	42	2	1	0	0
	L	0	25	0	0	0
	N	1	0	24	0	0
	P	0	0	0	25	0
	B	0	0	0	0	21

		classified as				
		H	**L**	**N**	**P**	**B**
true class	**H**	45	0	0	0	0
	L	1	24	0	0	0
	N	0	0	25	0	0
	P	2	0	0	23	0
	B	0	0	0	0	21

Table 1 shows the confusion matrices, i.e. how many examples of a certain true class (in rows) are classified in one of possible five classes (in columns).

For the real world cases, we can use confusion matrices for three purposes:

- We can observe how many false positives (false alarms) can be expected using these classifiers. When in real world use the system would report false alarm, e.g., normal walking is classified as some health problem, ambulance could drive to pick up the elderly which would cause unnecessary costs. In case of naive Bayes, normal walking was in 1 of 25 examples erroneously classified as hemiplegia.
- We can see how many false negatives can be expected using these classifiers. False negatives could mean potentially risky situation for the elderly, as his/her health problem would not be recognized automatically.

In case of naive Bayes, hemiplegia was in 1 of 45 examples erroneously classified as normal walking.

- We can identify between which health states (classes) the errors (misclassifications) occurs. Consequently, we can add additional features to help distinguish between those particular classes. The misclassifications happened very rarely.

The results show that in the proposed approach false positives/negatives are very rare, i.e., they would not cause much unnecessary ambulance costs. Since the method accurately classified most true health problems, it represents high confidence and safety for the potential use in elderly care.

4.1 Variation of Noise

To test the robustness of the approach, we added Gaussian noise with varying standard deviation (and zero mean) to the raw coordinates. The standard deviation of noise was varied from 0 mm to 50 mm in steps of 5 mm.

As a preprocessing step, a Kalman filter was used to smooth the potentially unrealistic difference between the positions of two consecutive time samples, caused by the addition of Gaussian noise to the captured positions [16]. The classification accuracies the classifiers, modeled on noisy data, are shown in Table 2.

Table 2. Classification accuracies (CA in [%]) of the naive Bayes (second row) and support vector machines classifier (third row) built on data with added noise. The table cells contain the classification accuracy and the significance level of the two-tailed paired t-test which was performed between the cross-validation folds of the initial setting (0 mm noise) and the corresponding amount of noise denoted by the table column. Dark shading denotes the experiments in which the classification accuracy significantly changed ($\alpha < 0.05$).

Noise [mm]	***0***	5	10	15	20	25	30	35	40	45	50
CA (NB)	***97.2***	97.9 α = 0.608	98.6 α = 0.168	97.2 α = 0.999	95.7 α = 0.443	95.7 α = 0.443	96.5 α = 0.654	95.7 α = 0.343	93.6 α = 0.096	92.2 α = 0.045	92.2 α = 0.068
CA (SVM)	***97.9***	98.6 α = 0.558	98.6 α = 0.558	97.9 α = 0.999	98.6 α = 0.343	98.6 α = 0.591	94.3 α = 0.049	95.0 α = 0.228	97.9 α = 0.999	90.8 α = 0.008	90.8 α = 0.004

Table 2 shows the results of varying the amount of standard deviation of noise in the range 0 – 50 mm. The results indicate that only significant decrease in classification accuracy happened at 45 mm for naive Bayes and at 30, 45 and 50 mm for support vector machines. Both algorithms manage to retain their performance long while increasing the data noise.

Even with the highest amount of noise added, the classification accuracies of both classifiers were much higher than the accuracy of the majority class classifier with classification accuracy of 31.9 %. Majority class classifier defines the lowest acceptable classification accuracy, which can be trivially achieved by classifying all examples into the majority class, i.e., class with the largest number of training examples. Therefore, the models are robust to the addition of the noise.

4.2 Reduction of the Number of Tags

Since wearing the full complement of 12 tags may be annoying to the user, we investigated ways to reduce the number of tags. We started with all 12 tags and removed them in the order that retained the largest number of the features when decreasing the number of tags by one. Consequently, the "best" tag placement for each number of the 12 tags was obtained. The classification accuracies for the best tag placements for various numbers of tags and without addition of noise are shown in Table 3.

The results show that decreasing the number of tags results in the decrease of classifier's performance. From Table 3 we can observe that 5 is the smallest number of tags for which the performance of classifiers is insignificantly different compared to the initial setting.

For recognizing target health states, the positions of tags from the most to the least appropriate are: shoulder, wrist, elbow, hip, ankle and knee.

Table 3. The classification accuracies achieved with the best tag placements for each number of tags. The table cells contain the classification accuracy and the significance level of the two-tailed paired t-test which was performed between the initial setting (12 tags) and the corresponding number of tags. Dark shading denotes the experiments in which the classification accuracy decreased significantly ($\alpha < 0.05$).

Nr. of tags	***12***	11	10	9	8	7	6	5	4	3	2	1
CA (NB)	***97.2***	97.9 $\alpha = 0.343$	98.6 $\alpha = 0.168$	97.9 $\alpha = 0.343$	97.2 $\alpha = 0.999$	92.9 $\alpha = 0.052$	96.5 $\alpha = 0.591$	93.6 $\alpha = 0.052$	92.9 $\alpha = 0.195$	83.0 $\alpha = 0.001$	68.8 $\alpha < 0.001$	50.4 $\alpha < 0.001$
CA (SVM)	***97.9***	99.3 $\alpha = 0.168$	98.6 $\alpha = 0.594$	98.6 $\alpha = 0.343$	98.6 $\alpha = 0.591$	95.0 $\alpha = 0.228$	99.3 $\alpha = 0.168$	92.9 $\alpha = 0.097$	87.2 $\alpha = 0.004$	75.2 $\alpha < 0.001$	59.6 $\alpha < 0.001$	46.1 $\alpha < 0.001$

4.3 Dependence of the Classification Accuracy on the Tag Placement and Noise Level

Besides experimenting with tag placement, for every placement we also varied the standard deviation of noise from 0 to 50 mm. Figure 1 shows the dependence of the classification accuracy (CA) on the tag placement and the noise level. We can observe a variation of the noise in the standard deviation from 0 to 50 mm on horizontal axis and the best tag placement for each number of tags from 12 to 1 tag on the vertical axis. Each curve of different color and shape (e.g. doted, dashed) connects points of the particular classification accuracy.

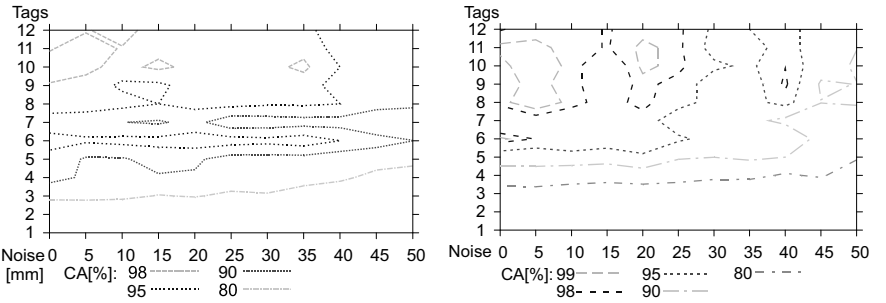


Figure 1. Classification accuracy with respect to the number of tags and the noise level for the recognition of the target health problems using the naive Bayes (left) and support vector machines classifier (right).

Figure 1 (left) illustrates that naive Bayes classifier never exceeded classification accuracy of 99 %. It (nearly) reached the classification accuracy of 98 % with 10-12 tags and 0-5 mm noise. At least 8 tags with 0-35 mm noise and precisely 6 tags with 0-40 mm noise were required to surpass the 95 % accuracy limit. At least 3 tags with 0-10 mm noise, 4 tags with 15-40 mm noise and 5 tags with 45-50 mm noise were needed to achieve the classification accuracy of over 80 %.

Figure 1 (right) shows the classification accuracies for the support vector machines classifier. With 8-10 tags and at most 0-5 mm noise or with 10-11 tags with around 20 mm noise we can achieve a classification accuracy of over 99 %. We need at least 8 tags and at most 0-10 mm of noise to surpass the boundary of 98 %, 6 tags with 0-25 mm noise for 95 % and 5 tags with 0-40 mm noise for 90 %. Four tags are enough to overcome the boundary of 80 % for all the noise amounts except for the 50 mm noise, where 5 tags are needed.

Although the classification accuracies of the support vector machines classifier are overall better than of naive Bayes, naive Bayes has important advantage in interpretability of the predictions. In case the system recognizes some health state it explains the reasons for classification to the physician.

5 Explanation of the Detection

Because it is very important for the physician to obtain an explanation for the interpreted health state, we also paid a lot of attention to it and developed a control-panel prototype, which is intended to be used by physicians (Figure 2).

In the middle of the prototype screen, there is visualization of the kinematic model of the elderly moving through the room. In the upper-right-hand corner there are controls for saving and loading the captured movement. Underneath there are the time series of the calculated angles.

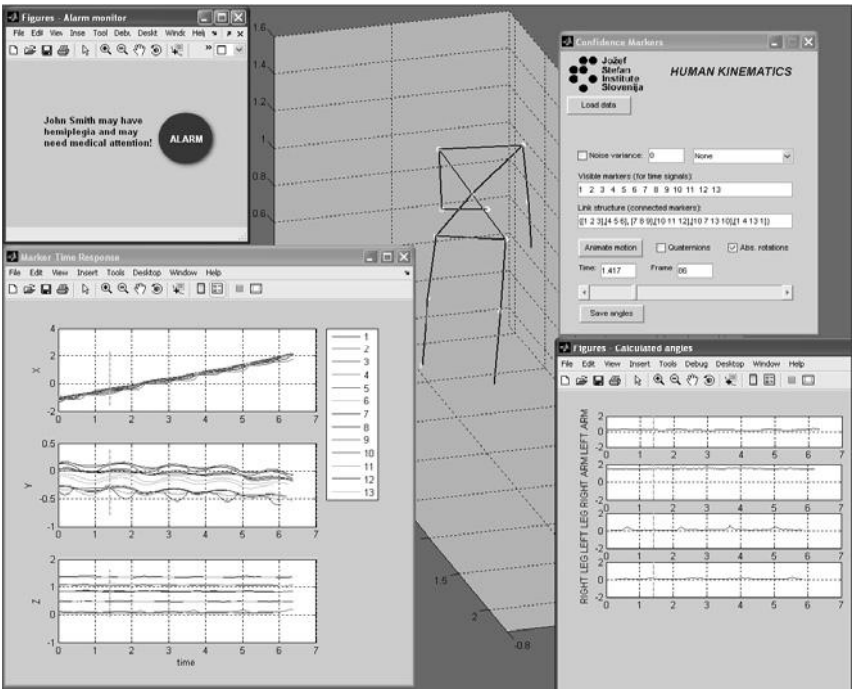


Figure 2. Panel prototype for the explanation of the interpreted health state.

In the lower-left-hand corner the time series of the x, y and z positions are shown for all 13 tags (the 13th tag is calculated from the positions of the nearest tags). When the health problem is recognized, the red alarm button appears in the left upper-left-hand corner with the necessary description of the recognized case for the relatives and for the medical center that has the control panel.

6 Conclusion

A system for detection of health problems through gait patterns of elderly to support their independent living is proposed in this paper. Time series of the detected positions of the body parts from the motion-capture system are transformed into a form for supervised machine learning methods using novel features for recognition of health problems. Although the automatic recognition of health problems is rarely addressed in research, our results are promising.

First objective of the research was to discover to what extent the automatic recognition of health problems with motion capture system is feasible. The results show that in the initial setting (no noise, all tags) naive Bayes and support vector machines algorithm achieved an average classification accuracy of 97.2 % and 97.9 %, respectively. This is much higher than the 31.9 % of the reference majority class; consequently, achieved classification accuracies are high. In the experimental results of initial setting, there were a few false positives/negatives, thus they would not cause much unnecessary ambulance costs in implementation of our approach in practice. Since the system accurately classified the most true health problems, it represents high confidence and safety for the elderly in the potential use in elderly care. Therefore, the automatic recognition of health problems with motion capture system is feasible to a high extent.

Second objective was to investigate the classification accuracy achievable using machine learning approach and various numbers/placements of tags on the user's body and various amounts of noise in tag coordinates. Tag placement must achieve a trade-off between usability and technical requirements – the users prefer as few tags as possible, but too few tags cannot ensure sufficient accuracy. Both the finding regarding noise and tag placement can affect motion-capture system selection and further development and applications of care systems for elderly.

The results of varying the amount of noise indicate that only significant decrease in classification accuracy happened at 45 mm for naive Bayes and at 30, 45 and 50 mm for support vector machines. Both algorithms manage to retain their performance long while increasing the data noise. The approach is therefore robust to the addition of the noise. The results of decreasing the number of tags show that it causes the decreasing of classifier's performance. Five is the smallest number of tags for which the classification accuracies of classifiers are insignificantly different compared to the initial one.

The results of investigation of the dependence of the classification accuracy on the tag placement and noise level show that support vector machines algorithm was much more accurate since it reached classification accuracy of over 99 % with only 8 tags with 0-20 mm noise, but naive Bayes was able to reach only 95 %. However, naive Bayes has important advantage over support vector machines, since the results of its classification are easily interpretable, i.e., if in practice the system would recognize some health state, the physician could check, why the system thinks it is certain health state.

Because it is very important for the physician to obtain an explanation for the interpreted health state, we also paid a lot of attention to it and developed a control-panel prototype for the explanation of the interpreted health state, which is intended to be used by physicians. When the health problem is recognized, the alarm button appears with the necessary description of the recognized case for the relatives and for the medical center that has the control panel.

For future work, we can modify the proposed system to be used also for automatic evaluation of rehabilitation process (e.g. after stroke) at home.

Acknowledgments

This work is partially financed by the European Union, the European Social Fund. The authors thank Martin Tomšič, Bojan Nemec and Leon Žlajpah for their help with data acquisition, Anton Gradišek for his medical expertise and Zoran Bosnić and Mitja Luštrek for helpful discussions.

References

1. D. Strle, V. Kempe, “MEMS-based inertial systems”, Informacije MIDEM 37(2007)4, pp. 199-209.
2. D. Jurman, M. Jankovec, R. Kamnik, M. Topič, “Inertial and magnetic sensors: The calibration aspect”, Informacije MIDEM 37(2007)2, pp. 67-72.
3. F. Dimic, B. Mušič, R. Osredkar, “An example of an integrated GPS and DR positioning system designed for archeological prospecting”, Informacije MIDEM 38(2008)2, pp. 144-148.
4. S. Ribarič, J. Rozman, “Sensors for measurement of tremor type joint movements”, Informacije MIDEM 37(2007)2, pp. 98-104.
5. J. Trontelj, J. Trontelj and L. Trontelj, “Safety Margin at mammalian neuromuscular junction – an example of the significance of fine time measurements in neurobiology”, Informacije MIDEM 38(2008)3, pp. 155-160.
6. Bourke, A.K., and Lyons, G.M. A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor. Medical Engineering & Physics 30, 1 (2006), 84–90.
7. Bourke, A.K., Scanaill, C.N., Culhane, K.M., O'Brien, J.V., and Lyons, G.M. An optimum accelerometer configuration and simple algorithm for accurately detecting falls. In Proc. BioMed 2006 (2006), 156–160.
8. Confidence: Ubiquitous Care System to Support Independent Living. http://www.confidence-eu.org.
9. Craik R., and Oatis C. Gait Analysis: Theory and Application. Mosby-Year Book (1995).
10. Perry J. Gait Analysis: Normal and Pathological Function. McGraw-Hill, Inc., 1992.
11. eMotion. Smart motion capture system. http://www.emotion3d.com/smart/smart.html.
12. Kangas, M., Konttila, A., Lindgren, P., Winblad, P., and Jamsa, T. Comparison of low-complexity fall detection algorithms for body attached accelerometers. Gait & Posture 28, 2 (2008), 285–291.
13. Lakany, H. Extracting a diagnostic gait signature. Pattern recognition 41 (2008), 1627–1637.

14. Luštrek M., Kaluža B., Fall Detection and Activity Recognition with Machine Learning. Informatica (Slovenia) 33(2): 197-204 (2009).
15. Maybeck, P.S. Stochastic models, estimation, and control. Mathematics in Science and Engineering 141 (1979).
16. Qian, G., Guo, F., Ingalls, T., Olson, L., James, J., and Rikakis, T. A gesture-driven multimodal interactive dance system. In Proc. ICME '04 (2004), 1579–1582.
17. Sukthankar, G., and Sycara, K. A cost minimization approach to human behavior recognition. In Proc. AAMAS 2005 (2005), 1067–1074.
18. Tapia, E.M., Intille, S.S., Haskell, W., Larson, K., Wright, J., King, A., and Friedman, R. Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor. In Proc. ISWC 2007 (2007), 37–40.
19. Vishwakarma, V., Mandal, C., and Sura, S. Automatic detection of human fall in video. Lecture Notes in Computer Science 4815 (2007), 616–623.
20. Witten, I.H., and Frank, E. Data Mining: Practical Machine Learning Tools and Techniques (2nd edition). Morgan Kaufmann (2005).
21. Zhang, T., Wang, J., Liu, P., and Hou, J. Fall detection by wearable sensor and One-Class SVM algorithm. Lecture Notes in Control and Information Science 345 (2006), 858–863.
22. Zouba, N., Boulay, B., Bremond, F., and Thonnat, M. Monitoring activities of daily living (ADLs) of elderly based on 3D key human postures. In Proc. ICVW 2008 (2008), 37–50.
23. Moore ST, et al., Long-term monitoring of gait in Parkinson's disease, Gait Posture (2006), doi:10.1016/j.gaitpost.2006.09.011

Selecting Features in Origin Analysis

Pam Green, Peter C.R. Lane, Austen Rainer, Sven-Bodo Scholz [1]

Abstract When applying a machine-learning approach to develop classifiers in a new domain, an important question is what measurements to take and how they will be used to construct informative features. This paper develops a novel set of machine-learning classifiers for the domain of classifying files taken from software projects; the target classifications are based on origin analysis. Our approach adapts the output of four copy-analysis tools, generating a number of different measurements. By combining the measures and the files on which they operate, a large set of features is generated in a semi-automatic manner. After which, standard attribute selection and classifier training techniques yield a pool of high quality classifiers (accuracy in the range of 90%), and information on the most relevant features.

1 Introduction

Classification models can be important tools for analysing large volumes of data. Machine-learning models can be even more valuable, as they are developed through labelled examples. However, the development of a classification model always assumes the prior existence of an adequate set of descriptive features or measurements of the data from which distinctions may be derived. The problem of constructing features from an initial set of measurements has been tackled in a variety of ways. For instance, Kramer [13] proposed the development of conjunctive features, using inductive logic techniques, and Krawiec [14] used genetic programming to develop functional extensions of the initial feature set. However, the problem of generating the initial set of features remains a challenge. Online feature selection, as used by Glocer [3] in image analysis, is one solution, building features on demand from a set of primitive filter results.

1 University of Hertfordshire, College Lane, Hatfield, Herts, AL10 9AB, UK.
p.d.1.green, p.c.lane, a.w.rainer, s.scholz@herts.ac.uk

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_29,

In this paper, we develop classification models for analysing the evolution of files in software projects. Rather like Glocer, we begin with a set of primitive measurements generated by some tools for analysing copying between files. Then, we develop larger sets of features, using a variety of functional and conjunctive combination techniques; some of these combinations are based on the original files, and some on the primitive measurements. Finally, this large set of features is used to develop classification models using standard feature selection and classification algorithms.

2 Application

Our application falls within the area of 'origin analysis', a term used in software evolution studies [4, 12] to describe tracing the source of new entities in a software system. Godfrey and Zou [4, p.1] state that "detecting where merges and splits have occurred can help software maintainers to better understand the change history of a software system" and "that having an accurate evolutionary history that takes structural changes into account is also of aid to the research community".

Files or functions which are renamed or moved can be relatively simple to trace. However, when these entities are split, merged or recombined, or when there are revisions such as the renaming of identifiers or the rearranging of code, the problem is more complex. Tracing this type of software restructuring involves matching features of the source code.

Previous approaches include finding the origin of methods by comparing combinations of call profiles, metrics, names and declarations [4]; and finding the origin of classes by vector space analysis of identifiers [2]. We have taken another approach, using the similarity measures provided by copy-analysis tools both to filter candidate restructured files from our database of source code, and to construct novel features which are used in classifying these candidates. In this paper, we focus on split files.

Similarity measures have been combined in software origin analysis before [12, 19], but not as the input to a machine-learning system. Our approach differs from previous studies in three ways. First, in the source of its features – in particular, in our use of the Ferret copy-analysis tool [15]. Second, we conduct a broader-scale search for relevant features, constructing a large initial set and then reducing the number. Third, we explore a range of classification algorithms, and present comparisons between them. In an earlier study [5], we showed the viability of our approach; in this paper we extend both the dataset and the feature set, and expand on the feature construction analysis.

3 Data Collection

There are four main stages in our analysis of software source code. The first step is downloading data from a repository. The second step is to filter sets of candidate items; while these sets contain the examples we want, they also contain other items. The third step is to construct the features. The fourth and final step is to classify the items. In spite of the wide availability of open-source software, there is a lack of marked-up datasets suitable for use directly in data-mining studies (e.g. [11]). For this project, we have constructed our own dataset by manually classifying a set of candidates to build the classifiers.

Our experiments explore the effect of using different combinations of features in the classification process. We focus on C code in this investigation, however, our method can be simply adapted for other programming languages and can be applied to any evolving text-based data.

3.1 Collection and Preprocessing

The source code selected for this study was drawn from the open-source repository SourceForge [17]. The 84 projects are taken from different domains. The chosen projects were downloaded automatically, releases determined by the project authors' tags, and each release placed in a separate, numbered, sub-directory. Our database contains over 266,000 C code files. Comments were removed from the files, leaving approximately 70 million lines of code.

3.2 Filtering

We employed Ferret for comparing files in the filtering stage. Ferret computes a similarity score between files based on the trigrams in them; the score ranges from 0 (no similarity) to 1 (identical). The advantage of Ferret is its efficient processing time, which is approximately linear in the amount of input code [15]. We define terms used in the remainder of this paper in Table 1.

The first step in filtering is to store vectors of similarities between files in each project. The three data sets are: **self**-similarity, of each file to the same file in the next release; **cons**ecutive-similarity, of each file to every other file in the next release; and **same**-similarity, of each file to every other file in the same release. Vectors of file sizes in bytes are also stored for use in filtering.

Figs. 1 and 2 illustrate similarities for an example mini-project. This example has 3 releases and 2 original files, a and b. File b is split to form file c before release 2. Fig. 1 lists similarities, with a selection shown in Fig. 2.

Self-similarity shows whether a file has changed between releases. Consecutive- and same-similarity are used to detect both similarity between files and changes to

Table 1 Terminology

Term	Notation	Description
Candidate file	f_a^n	Any file a in release n, matching filter criteria
Amended file	f_a^{n+1}	Revision to file a in the next release
New file	$f_b^{n+1} \bullet \nexists f_b^n$	A file in release n+1 not in release n
Disappearing file	$f_a^n \bullet \nexists f_a^{n+1}$	A file in release n not in release n+1
Similarity	$\mathrm{Sim}(f1, f2)$	Jaccard coefficient of similarity between files f1 and f2, based on token trigrams
Consecutive-sim'y	$Sim(f_x^n, f_y^{n+1})$	Between 2 files in consecutive release
Self-similarity	$Sim(f_a^n, f_a^{n+1})$	Consecutive similarity of files of same name
Same-similarity	$Sim(f_x^n, f_y^n)$	Between files in the same release
Similar file	f_x^{n+1}, $Sim(f_a^n, f_x^{n+1}) > \text{Min-sim}$	File in release n+1 with a similarity to the candidate file above the minimum threshold

similarity between releases. In the example, file a is unchanged between releases 1 and 2 (1.0), and has minor edits before release 3 (0.95); file b splits after release 1 (0.6), forming similar file c (0.4).

The second step in filtering involves scanning the stored vectors. Filtering differs between applications. For example, for renamed or moved files, the main filter is based on self-similarity, to find files which exist in one release, n, but not in the next, n+1. Consecutive-similarity provides a second filter.

Split file filters are based on size and similarity. A split file usually becomes smaller; and the self-similarity is normally lower than would be expected after minor edits. Also one or more files will increase their similarity to the old file.

Candidate split files are grouped with files from the next release to form a *comparison group*. These files consist of the *amended* version of the candidate file, and *target* files, which can be any mix of new files similar to the candidate, and existing files which have become more similar to the candidate than they were in the previous release. There can be one or more target files in a group.

The mean consecutive-similarity across all projects in this study is around 0.04. This figure excludes self-similarity, where the mean is 0.95. Candidate files for this experiment were filtered with: 10% minimum size change; 0.9 maximum self-similarity; and 0.1 minimum consecutive-similarity.

same a-b <0.07, 0.02, 0.03>
b-c <-1, 0.15, 0.13>
a-c <-1, 0.02, 0.01>
self a <1.0, 0.95>
b <0.6, 0.96 >
c <-1, 1.0>
cons a-b <0.02, 0.01>, b-a <..>
a-c <..>, c-a <..>
b-c <0.4, 0.37>, c-b <..>

Fig. 1 Example similarity vectors

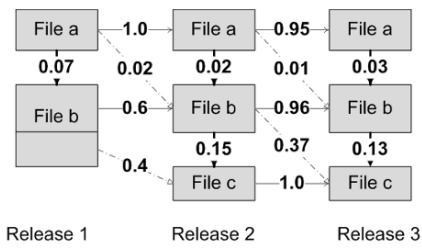


Fig. 2 Selected similarities shown

Table 2 Analysis of file type and classification of candidate files

File type	.c	.h	Total
Split	79	58	137
Not	112	39	151
Total	191	97	288

3.3 Experimental Data

The class of each candidate file was decided by visual inspection of all files in the comparison group. Two example split files are shown in Figs. 3 and 4. As this dataset is collected at release level, most instances are more complex than these examples. Blocks of code from candidate files can be more entangled with new or existing code; files are often subject to several revisions; and there may be multiple target files, for example, one of our files is split into 8 of its 12 target files. Some files selected by filtering had so many revisions that they could not be classified; these examples were excluded.

Table 2 shows the composition of the resulting dataset. Of the 288 instances, 137 are classed as split files (79 .c and 58 .h); this provides an almost balanced dataset with 48% positive and 52% negative split file examples. File sizes vary from 5 to 9377 lines and there are between 1 and 15 target files in the comparison groups.

Filtering parameters are set with the aim of selecting all relevant files. This means that other files which happen to share similar qualities are also selected. To discriminate between the files of interest and other files which match the filter criteria, we construct a set of features which are used to train a machine-learning system to classify the files.

Features constructed for this task are based on information which might be used by someone trying to determine whether a file belongs to a certain category. For example, in deciding whether a file is split, natural questions may include: Are blocks of code missing from the candidate file? Do all or most of these blocks appear in other files? Were they already in those files? How many other files share these blocks? How similar are the associated files? Are the blocks large enough to be interesting? What proportion of the file is copied? How is the copied code distributed within the files?

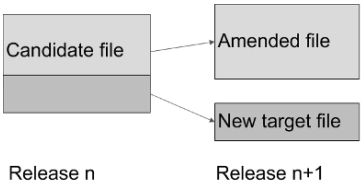


Fig. 3 A simple split file

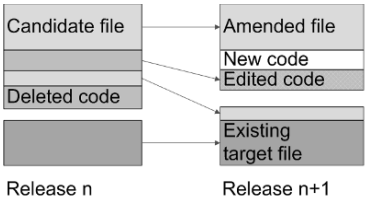


Fig. 4 A more complex split

Knowing how best to unpack these questions and find suitable quantities to measure is problematic. We adopt a generate and select approach, creating a large pool of features using a range of measurements and sources; then apply a feature selection algorithm to the pool to discover an effective subset.

4 Feature Construction

The features constructed for this system are based on comparisons between the files in a group. Two different bases for comparison are used. First, pairwise comparisons between the candidate file, the amended file and the target file with the largest increase in similarity to the candidate file. The second set of comparisons are between the candidate file and various concatenations of the amended file and the target files. These combinations are detailed in Sect. 4.3. We use four tools, described in Sect. 4.1, to compare the files. The features based on each tool's similarity measures are discussed in Sect. 4.2.

4.1 Tools

In addition to its use in filtering the files, Ferret is used for feature construction. To this we add three code clone detection tools: Duplo [1], Code Clone Finder (CCFinderX) [10] and Simian [9]. Clone detection tools seek identical or similar sections of code in a set of files. The tools use one or more of five broad categories of code representation: text, tokens, abstract syntax trees, program dependency graphs or metrics [16]. They employ different matching algorithms and offer a range of parameterisation options.

The reorganised code in restructured files can be difficult for simple algorithms to match. The variety and flexibility offered by clone detection tools can overcome some of these difficulties, such as renamed identifiers, and rearranged or edited code. We have chosen to use text- and token-based detection methods which require little preprocessing. We post-process the detected clones to approximate a one-to-one correspondence of copied code in the files [7]. The tools we combine use complementary matching algorithms.

Ferret detects copying in text and program code. The input source code is tokenised, making whitespace and formatting irrelevant. Efficiency is achieved by using a one-pass algorithm. Several forms of output are provided, including a list of pairwise similarity scores, with summary trigram information for the files; a list of all trigrams, and the files in which they occur; and an XML report which separates blocks of code covered by matched trigrams from blocks with no matching code in the other file. These three outputs are analysed, the XML report in two different ways, to provide four sets of features based on the relationship between files in a comparison group.

Duplo compares files to find blocks of matching lines. A block is defined by two parameters: minimum characters per line and minimum consecutive matched lines per block. Code is prepared by removing whitespace and, optionally, preprocessing directives. Significant lines (those containing the minimum number of characters specified) are hashed to optimise matching. A matrix, coded to show the matches, is scanned diagonally to find blocks of at least the minimum number of consecutive matching lines of code.

CCFinderX is a token-based clone detector. Preprocessing filters out parts of the code likely to produce false clones because of their repetitious structure, such as table initialisations. The remaining code is tokenised, transformed and simplified using language-specific rules, producing a sequence of key words and parameterised identifiers. To detect renaming, we use one-to-one parameter matching. Clones are specified by a minimum number of matching tokens, and of different token types in the sequence.

Simian lies between Duplo and CCFinderX, in that matching is based on hashed lines of transformed code, with tokenisation used to offer parameterisation options. We use parameters to ignore literal case, identifiers and modifiers. A minimum number of significant lines specifies clone size.

4.2 Measurements

Output from the four tools is analysed to provide eight different sources of features, which are listed in Table 3. In this section, descriptions of the sources are grouped according to the amount of processing required to generate features from the source. The majority of measures are different for each of the files being compared; for example, the distribution of copied blocks differs between files. These measures are therefore recorded for each file.

4.2.1 Ferret and Duplo Basics

Features in the basic sets are taken directly from tool outputs. Ferret's default output lists the number of trigrams in each file and shared by files, the similarity score, and the containment of each file in the other.

Table 3 Keys and names of the feature sources

FB	Ferret Basics	DB	Duplo Basics
FT	Ferret Trigrams	DU	Duplo Blocks
FX	Ferret XML Blocks	CC	CCFinderX Blocks
FD	Ferret XML Density	SM	Simian Blocks

Duplo outputs line numbers for the start of each matching block, the code in the block, and summary figures which include the total number of lines in the files, the number of duplicated blocks and duplicated lines.

4.2.2 Ferret Trigrams

The Ferret trigram list is analysed to discover relationships between the files in a group. For example, a candidate file which is subject to a simple two-way split (see Fig. 3) will have a large proportion of its trigrams present in one or other of the files resulting from the split; however, these two files should share only a few incidentally similar trigrams. Also, few trigrams should be unique to any individual file. In a multiway split, most candidate file trigrams should appear in a concatenation which includes the resulting files.

As files differ in size, it is difficult to decide whether the number of trigrams, or the proportion of the file they represent, is likely to be the better indicator of the relationship between files. We include both measures, for example, the number of trigrams common to the candidate file and the target file; or the proportion of trigrams from one file shared with another file.

4.2.3 Block Based Measures

The five remaining feature sets have much in common. Each is constructed from information about clones or blocks provided by one of the four different tools. Duplo and Simian report on line-based clones, Duplo giving exact matches and Simian parameterised matches; CCFinderX reports on tokens in filtered, transformed and parameterised code; and Ferret relates shared trigrams to the source code. The Ferret report is analysed in two ways: to find both contiguous blocks of copied code, and densely copied blocks [6].

While the information from the tools differs in detail, there are two common problems. One is choosing a suitable set of parameters to produce meaningful output from comparisons between files. Whichever settings are used for the tools, the other problem is that of describing the resulting set of blocks and their sizes to provide useful information about the copying between files.

Duplo, CCFinderX and Simian allow minimum clone size to be specified; however, deciding on a size is difficult. If too small, trivial clones will be detected; if too large, then meaningful clones may be missed. This problem is compounded by the varying file sizes. We select a range of absolute and proportional sizes, running the tools with each set of parameters.

Clone detection tools and Ferret XML analyses provide information about each copied block in a file. These blocks vary in number and content between files. For a machine-learning application, information must be presented as a standard-sized vector for each file. Deciding how to do this is a problem. It could simply, but not informatively, be described with one figure, such as a block count, or total lines,

Table 4 The 2972 features. The upper part of the table shows the files used in pairwise comparisons (1–3) and the concatenations compared with the candidate file (A–F). The lower part of the table shows the source keys (see Sect. 4.1), the number of features for each type of comparison, names and units measured for each source.

Files compared pairwise	1	Candidate and amended files
	2	Candidate and target file most similar to the candidate
	3	Amended and target file most similar to the candidate
Concatenations of files compared to candidate	A	Amended and all target files
	B	All new files
	C	The 2 files with the highest similarity to the candidate file
	D	Amended and all new files
	E	Amended and target file with highest similarity to candidate
	F	Amended and new file with highest similarity to candidate

Key	No.Features Pairs	Conc.	Source name and outline measurements used in features
FB	21	42	Ferret Basics: similarity, containment, ratio of file trigrams
FT	27	54	Ferret Trigrams: trigrams in 1, 2 or 3 files
FX	126	252	Ferret XML: blocks, lines, tokens, characters; range of block sizes
FD	240	480	Ferret XML Density: dense blocks, lines; 2 density parameter sets
DB	39	78	Duplo Basics: blocks, lines; range of parameters
DU	204	408	Duplo Unscrambled: matched blocks, lines; range of parameters
CC	138	276	CCFinderX: blocks, tokens; 2 parameter sets for each file type
SM	195	390	Simian: blocks, lines; range of parameters
SS	2		Sundries: file type (c or h), number of target files in the group

trigrams or tokens in the blocks, or their ratio to file size. However, more information is conveyed by choosing a range of sizes or proportions, and describing the blocks in each range. The ranges can be reported individually or cumulatively. Again, we measure in various ways and leave assessment of the utility of the features to a selection algorithm.

4.3 The Feature Set

Table 4 provides information about the feature set. The top section of the table shows the comparisons on which the features are based. Within this, single file comparisons are numbered 1–3, and the different concatenations, which are compared to the candidate file, are marked A–F. The lower section of the table shows the key on the left and number of features from each comparison type in columns 2 and 3. On the right are source names and measurements used in feature construction. The concatenations combine the target files in different ways, with or without the amended file. For example, we combine the amended file and all target files in a group (A); or all of the new target files (B). Grouping different combinations of target files allows the same number of comparisons, and therefore the same number of features, to be built for each instance, regardless of the varying numbers of target files.

5 Experiments

The aim of these experiments is to compare the performance of different classification algorithms in split file identification using the features described in Sect. 4. We also explore the effect of feature selection on the training sets prior to building and testing classifiers.

5.1 Method

We applied machine-learning algorithms to classify the dataset using both the full feature set, and separate sets constructed from each source; for example, sets based only on analysis of the Ferret trigram report (FT). We also subdivide each set, creating subsets of the features built either from single file comparisons, or from concatenated file comparisons.

The full feature set was used to classify the data with 34 algorithms chosen from the Weka toolbench [18]. To test generalisation, each of the selected learning algorithms was used to build models with 100 selections of 66% of the dataset, leaving 34% for testing. We recorded the mean percentage of correct classifications of the test sets for each classifier with each feature set.

Five algorithms giving the best correct classification rates on the full feature set were selected. These five and their three-member subsets were combined with Grading, Stacking and Voting algorithms. The best combination, a majority vote between the five, was added to the chosen algorithms. The resulting set of six algorithms was used to test the classification rates of the subsets of features constructed from the eight different sources (see Table 3).

We also tested feature selection using the Attribute Selected Classifier (ASC) to apply the Correlation-based Feature Subset Selection algorithm (CFSS) with best-first search [8] to each training set before building and testing classifiers using the same six algorithms.

5.2 Results

Results for the six chosen algorithms are reported for the full feature set, and for sets using only single file comparisons. Features based solely on concatentation comparisons do not classify as well and therefore are not reported.

The full feature set results are recorded on the left in Table 5. Classification rates for ASC are shown in the third column, with the difference between the two in the next column. The same pattern is repeated in the next three columns for the set of features built from only the pairwise comparisons of single files. The last 2 columns show the differences between the two sets; for all features in the sets, (col.2 - col.5), and for ASC, (col.3 - col.6).

Table 5 Mean classification rates for selected algorithms. Results are grouped by feature set: the full feature set (built from comparisons between both single files and concatenations) in columns 2–4; with single file features in columns 5–7. The algorithms are applied to the training sets in two ways: to every feature (All), and to features selected from each training set by CFSS using ASC (ASC); their difference is also shown. Cols. 8 and 9 show the differences between full and single feature sets.

	Full feature set			**Single file features**				
	All	ASC	Diff	All	ASC	Diff	ΔAll	ΔASC
Simple Logistic	88.30	88.60	0.30	88.49	87.83	-0.66	-0.19	0.77
SMO	88.35	88.20	-0.15	87.79	88.07	0.28	0.56	0.13
Decorate	88.97	88.87	-0.10	88.48	88.31	-0.17	0.49	0.56
Rotation Forest	88.72	89.33	0.61	88.39	88.76	0.37	0.33	0.57
Random Forest	88.19	89.69	1.50	88.50	89.40	0.90	-0.31	0.57
Vote	89.40	89.76	0.36	89.22	89.08	-0.14	0.18	0.68

Features from each source introduced in Sect. 4.2 were used to train the classifiers. Classification rates are shown for the majority vote classifier, as a representative algorithm, in Table 6. Columns 3–5 show results for the source with features built from all comparisons, and columns 6–8 with features from single file comparisons. The results in columns 3–10 follow the pattern in Table 5, columns 2–9.

Table 6 Mean classification rates of a majority vote between the 5 algorithms for each source of features. The results are laid out in the same way as those in Table 5.

		Full feature set			**Single file features**				
Key	**Source**	All	ASC	Diff	All	ASC	Diff	ΔAll	ΔASC
CC	CCFinderX	77.99	73.44	-4.55	78.81	71.73	-7.08	-0.82	1.71
SM	Simian	84.44	82.67	-1.77	84.21	81.74	-2.47	0.23	0.93
DU	Duplo Blocks	85.37	83.31	-2.06	84.87	83.62	-1.25	0.50	-0.31
DB	Duplo Basics	88.18	86.98	-1.20	88.75	87.89	-0.86	-0.57	-0.91
FB	Ferret Basics	87.18	86.31	-0.87	88.37	85.30	-3.07	-1.19	1.01
FD	Ferret Density	88.79	87.72	-1.07	88.96	86.84	-2.12	-0.17	0.88
FT	Ferret Trigram	88.46	88.66	0.20	89.13	88.72	-0.41	-0.67	-0.06
FX	Ferret Blocks	89.58	89.76	0.18	89.94	88.96	-0.98	-0.36	0.80

6 Discussion and Summary

The process of constructing features from primitive measurements described above has been successful: the classification models generally perform well, achieving 88–89% accuracy ($\pm$ 2.6–3.2), a majority vote marginally improving the result. A natural question is which of the measurements and their sources were most important in constructing classifiers: Table 7 shows us that every source and all kinds of comparison were present in the set selected by CFSS.

Table 7 The features selected from the full set by CFSS. The source key is in the left-hand column, a prepended C shows concatenation comparisons. The next two columns show the keys of the files in the comparison. Column 4 shows on which of these files the measurement is based. Feature source and file keys are in Table 4. In column 5 a tick means that the measure is proportional, and an asterisk shows that high values for this feature tend to indicate negative instances. The measure is described in the last column.

Source	Compared		Measured	Ratio	Brief description of measure
FB	2	3	either	✓∗	Similarity score
FT	1	2 or 3	1	✓	Trigrams common to 1 ∧ (2 XOR 3) to total
FT	1	3	1	✓	Common trigrams to total
FT	1	3	3	✓	Common trigrams to total
FT	2	1 and 3	2	✓∗	Trigrams common to all 3 files to total
FT	3	1 and 2	3	✓∗	Trigrams common to all 3 files to total
FX	1	3	1	✓	Lines in blocks $\geq$ 16 lines to total
FX	1	3	1	✓	Lines in blocks $\geq$ 8 lines to total
FX	1	3	3	✓	Lines in blocks $\geq$ 8 lines to total
FX	2	3	2	✓∗	Blocks $\geq$ 8 lines to all blocks
FD	1	3	1	–	Mean copied tokens in blocks $\geq$ 0.95 density
FD	1	3	1	✓	Size of largest block $\geq$ 0.95 density to total
FD	1	3	1	–	Mean no. all tokens in blocks $\geq$ 0.95 density
FD	2	3	2	✓∗	All tokens in blocks $\geq$ 0.70 density to total
DB	1	3	1	✓	Lines in blocks $\geq$ 4 lines to total
DB	1	3	1	✓	Lines in blocks $\geq$ 8 lines to total
DU	2	3	3	✓∗	Lines in blocks $\geq$ 2 lines to total
CFX	1	A	1	✓	Copied to total characters
CFX	1	D	1	✓	Blocks $\geq$ 4 lines to all blocks
CFX	1	E	1	✓	Copied to total characters
CFX	1	E	E	✓	Blocks $\geq$ 4 lines to all blocks
CFD	1	A	1	✓	Tokens in 0.7$\leq$ dense blocks $\geq$ 100 tkns : total
CFD	1	C	C	✓	Size of largest block $\geq$ 0.95 density to total
CFD	1	C	C	✓	Size of largest block $\geq$ 0.70 density to total
CDD	1	C	1	✓	Lines in blocks $\geq$ 2 lines to total
CDU	1	C	C	✓	Lines in blocks $\geq$ 16 lines to total
CDU	1	E	E	✓	Lines in blocks $\geq$ 2 lines to total
CSM	1	B	B	✓	Number of blocks $\geq$ 16 lines to total
CSM	1	E	E	✓	Number of blocks $\geq$ 4 lines to total
CSM	1	F	1	–	Number of clones $\geq$ 8 lines
CCC	1	C	C	✓	Tokens in blocks $\geq$ 25/6 (.c/.h) tokens to total
CCC	1	E	1	✓	Tokens in blocks $\geq$ 200 tokens to total

Selected Ferret trigram features include those measuring relationships between both pairs and triples of the singly-compared files. Many Ferret basic features are related to the filtering criteria; the one feature selected from this source is a similarity score not used in filtering. Only concatenation-based features are selected from the Simian and CCfinderX sets. Features from the four remaining sources are evenly divided between the two comparison bases.

The concatenations prove useful: the selection includes at least one feature based on each concatenation. Those involving either one or two files with the highest similarity to the candidate file (C and E) predominate; these features counteract our bias

towards new files in selecting the principal target file. Two of the features involve all of the new files (concatenations B and D); we have noticed that when a large file is split into multiple files, it is usually during the early stages of a project, when target files are likely to be new.

Features selected from each of the source groups follow a similar pattern to the selection from the full set. Single file and concatenation comparisons are equally represented, and a range of files are compared; the majority of single file comparisons quantify the similarity between candidate and target files, and a minority between target and amended files; measurement bases are mixed; and most selected features are proportional, with a few absolute measures in each set. The majority vote classifiers based on the Ferret XML block feature set (88.96–89.94%±2.9–3.2) have similar classification rates to those based on the full feature set (89.08–89.76%±2.7–3.0).

A surprising finding is that one feature correctly classifies 86.46% of the dataset. This is the proportion of trigrams from the candidate file which are shared with *only* the target file. With over 9.3% of the candidate file trigrams in the target file, classification is positive, otherwise it is negative; in this way, 80% of the positive and 95% of the negative classifications are correct.

The feature construction process we have adopted is related to previous work in this area in several ways. Rather like Glocer [3], we construct features based on more primitive measurements. However, we construct a large group of features from our data in a batch, rather than using an online approach. Glocer's application was image analysis, in which pixel values are to some extent dependent on their neighbours; this fact is exploited to develop edge features. Similarly, in our application, files are related to one another temporally and by textual similarity, and we exploit these relations to narrow the range of potential constructed features. After building the initial features, we use a feature construction process related to those of Krawiec [14] and Kramer [13] mentioned earlier, except we use direct methods of combining features to construct new ones. This approach is fairly automatic, and CFSS then takes on the task of generating a robust set of features from which the final classification models are developed.

In conclusion, we have demonstrated that a technique of combining raw measurements to construct features in a fairly direct way supports the development of a robust set of classifiers and have identified effective sets of features. In particular, the value of using alternative sources of measurement is shown by the set of selected attributes containing examples from all sources. The classifiers have been used to develop reliable models of split files within the domain of software origin analysis. In future work, it would be interesting to explore further automation of the feature construction process, and also to work on a wider set of classification tasks within software evolution.

References

1. Ammann, C.M.: Duplo - code clone detection tool. Sourceforge project (2005) http://sourceforge.net/projects/duplo/
2. Antoniol, G., Penta, M.D., Merlo, E.: An automatic approach to identify class evolution discontinuities. In: IWPSE '04: Proceedings of the Principles of Software Evolution, 7th International Workshop, pp. 31–40. IEEE Computer Society, Washington, DC, USA (2004)
3. Glocer, K., Eads, D., Theiler, J.: Online feature selection for pixel classification. In: ICML '05: Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany (2005)
4. Godfrey, M.W., Zou, L.: Using origin analysis to detect merging and splitting of source code entities. IEEE Trans. Software Eng. **31**(2), 166–181 (2005)
5. Green, P., Lane, P.C.R., Rainer, A., Scholz, S.B.: Building classifiers to identify split files. In: P. Perner (ed.) MLDM Posters, pp. 1–8. IBaI Publishing (2009)
6. Green, P., Lane, P.C.R., Rainer, A., Scholz, S.B.: Analysing ferret XML reports to estimate the density of copied code. Tech. Rep. 501, Science and Technology Research Institute, University of Hertfordshire, UK (2010)
7. Green, P., Lane, P.C.R., Rainer, A., Scholz, S.B.: Unscrambling code clones for one-to-one matching of duplicated code. Tech. Rep. 502, Science and Technology Research Institute, University of Hertfordshire, UK (2010)
8. Hall, M.A.: Correlation-based feature subset selection for machine learning. Ph.D. thesis, University of Waikato, Hamilton, New Zealand (1998)
9. Harris, S.: Simian. http://www.redhillconsulting.com.au/products/simian/. Copyright (c) 2003-08 RedHill Consulting Pty. Ltd.
10. Kamiya, T., Kusumoto, S., Inoue, K.: CCFinder: a multilinguistic token-based code clone detection system for large scale source code. IEEE Trans. Softw. Eng. **28**(7), 654–670 (2002)
11. Kim, M., Notkin, D.: Program element matching for multi-version program analyses. In: MSR '06: Proceedings of the 2006 international workshop on Mining software repositories, pp. 58–64. ACM, New York, NY, USA (2006)
12. Kim, S., Pan, K., Jr., E.J.W.: When functions change their names: Automatic detection of origin relationships. In: 12th Working Conference on Reverse Engineering (WCRE 2005), 7-11 November 2005, Pittsburgh, PA, USA, pp. 143–152. IEEE Computer Society (2005)
13. Kramer, S., de Raedt, L.: Feature construction with version spaces for biochemical applications. In ICML '01: Proceedings of the 18th International Conference on Machine Learning, (2001)
14. Krawiec, K.: Genetic programming-based construction of features for machine learning and knowledge discovery tasks. Genetic Programming and Evolvable Machines **3**, 329–343 (2002)
15. Rainer, A.W., Lane, P.C.R., Malcolm, J.A., Scholz, S.B.: Using n-grams to rapidly characterise the evolution of software code. In: The Fourth International ERCIM Workshop on Software Evolution and Evolvability (2008)
16. Roy, C.K., Cordy, J.R., Koschke, R.: Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. Sci. Comput. Program. **74**(7), 470–495 (2009)
17. Sourceforge open source software repository : http://sourceforge.net/ (1998)
18. Witten, I.H., Frank, E.: Data mining: Practical machine learning tools and techniques with Java implementations. Morgan Kaufman, San Francisco, CA, USA (2000) http://www.cs.waikato.ac.nz/ml/weka
19. Yamamoto, T., Matsushita, M., Kamiya, T., Inoue, K.: Similarity of software system and its measurement tool SMMT. Systems and Computers in Japan **38**(6), 91–99 (2007)

AI FOR SCHEDULING

An Extended Deterministic Dendritic Cell Algorithm for Dynamic Job Shop Scheduling

X.N. Qiu[1] **and H.Y.K. Lau**[2]

Abstract The problem of job shop scheduling in a dynamic environment where random perturbation exists in the system is studied. In this paper, an extended deterministic Dendritic Cell Algorithm (dDCA) is proposed to solve such a dynamic Job Shop Scheduling Problem (JSSP) where unexpected events occurred randomly. This algorithm is designed based on dDCA and makes improvements by considering all types of signals and the magnitude of the output values. To evaluate this algorithm, ten benchmark problems are chosen and different kinds of disturbances are injected randomly. The results show that the algorithm performs competitively as it is capable of triggering the rescheduling process optimally with much less run time for deciding the rescheduling action. As such, the proposed algorithm is able to minimize the rescheduling times under the defined objective and to keep the scheduling process stable and efficient.

1 Introduction

The classic job shop scheduling problem (JSSP) assumes that a known set of jobs are to be scheduled on a series of machines and the task is to arrange them in a sequence to optimize a particular performance objective [1]. All the information is known and remains unchanged before the process starts. This type of problem is generally known as static JSSP. This problem has been well solved by a wide variety of approaches. A comprehensive survey about these techniques has been presented in [2]. However, many real world problems are dynamic, always subject to random perturbations, such as machine breakdowns and new job releases.

[1] Industrial and Manufacturing Systems Engineering Department, The University of Hong Kong, Hong Kong, P.R. China
shirley-qiu@hku.hk

[2] Industrial and Manufacturing Systems Engineering Department, The University of Hong Kong, Hong Kong, P.R. China
hyklau@hku.hk

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_30,

Often, a significant portion of information is uncertain or stochastic. All these situations are very common in the real world and are considered as dynamic JSSP.

There are two main directions in solving dynamic JSSP in the field of research. One direction is to deal with the situation that most of the information in the job shop is unknown and stochastic, and the events occurred during the process are unpredictable, which can be regarded as the continuous disturbances. The most representative case is that the random and prior unknown jobs arrive continuously over time [3]. This problem is very popular and well-studied. There are a great number of studies in the literature, which can be classified into three types [4]: simulation-based approach: such as the dispatching rules; AI and knowledge based approaches: such as GA, neural networks, fuzzy logic, expert systems; and agent-based approaches. Although the academic contribution to job shop scheduling is abundant, its impact on practice is minimal as most of the methods are not mature enough to apply in the real job shop [5]. The most popular and preferred approach in the industry is dispatching rules. A review of this method used in JSSP is presented in [6]. This approach is relatively robust to variability and uncertainty. However, a major criticism against dispatching rules is that there is no single universal rule that can consistently perform better than the rest in all possible situations that the shop may be in, because the performance of the dispatching rules is highly affected by the job shop configuration, scheduling measurements of interest, and the job shop state at each moment. Since the scheduling environment itself is dynamic, the dispatching rule needs to change over time according to the current nature of the job shop. Though, this is not the focus of this paper. Another direction of dynamic JSSP deals with the sudden disturbances occurred occasionally in the pre-arranged scheduling process. Most of the information is known with some unexpected changes. With regard to this situation, rescheduling — the process of updating the ongoing processing schedule in response to disruptions or other changes [7], is commonly used to minimize the impact when an event causes significant deterioration to the working system. There are mainly three kinds of rescheduling strategies, namely, event-driven rescheduling approach, periodic rescheduling policy and their hybrid [7]. The event-driven rescheduling approach is to reschedule the system once it runs into an unexpected event, and the periodic policy is to reschedule the process after the pre-defined interval. Jang [8] adopted the event-driven rescheduling approach and proposed a heuristic method based on a myopically optimal solution to construct a new schedule as soon as the stochastic jobs arrived. The result shows that the performance is unsatisfactory as constantly changing production schedule causes instability of the system and increases the related cost. Rangsaritratsamee [1] applied a genetic local search algorithm to the periodic rescheduling strategy that the new schedules were generated at each rescheduling point. This method increases the system's stability, but it endures the difficulty of finding the best rescheduling interval. If the interval is too short as of event-driven rescheduling policy, the system becomes unstable. On the other hand, an overly long interval will delay the emergent events handling that may cause great loss. Therefore, both

rescheduling methods have limitations. Hence a hybrid method is proposed, though an efficient means to hybridize the two methods remains a hard problem. As a result, it is very difficult to decide to how to use the rescheduling strategy. Due to its inherent complexity, priority dispatching rules are also commonly used in this case because of their simplicity and computational efficiency though they do not aim at achieving the exact solutions. Meta-heuristics, like neural network and genetic algorithm, are also adopted. More explicit review of techniques for dynamic JSSP can be found in [9]. In summary, most of the approaches are based on priority dispatching rules which is simple but not sufficient to achieve the optimal results. Although there are some other more efficient methods proposed recently, like meta-heuristics, most of them only consider one kind of disruption. However, in a practical system, different kinds of emergent events occur continuously. Therefore, complex dynamic JSSP is still not well solved and there exists much room for research. As such, this paper focuses on the second situation of dynamic JSSP and uses artificial immune systems related theories to deal with various types of disturbances to decide how to derive the rescheduling plan.

Artificial Immune Systems (AIS) is a relatively new theory. Inspired by the human immune system, AIS provides many appealing features which makes it unique from other evolutionary algorithms, including self-learning, long lasting memory, cross reactive response, discrimination of self from non-self, and strong adaptability to the environment [10]. These specific features are considered highly suitable and applicable in dealing with dynamic JSSP, but there are relatively few literatures reported on AIS-based techniques and algorithms in this area. Recently, a new breed of AIS, called Dendritic Cell Algorithm (DCA) [11], is proposed and shows success in solving computer security problems, which mainly involves the detection of intrusion and scanning port attacks. Similarly, in a typical dynamic JSSP, DCA can be adopted because unexpected events in a dynamic job shop can be regarded as the occurrence of intrusions while the job shop system corresponds to the computer system. Such parallels suggest that AIS-based algorithms would be strong candidates for providing solutions to dynamic JSSP.

This paper aims at designing and implementing an extended deterministic DCA based on the original DCA to arrange the rescheduling process for different kinds of sudden disturbances in dynamic JSSP. The paper is structured as follows: Section 2 defines the dynamic JSSP of interest. In Section 3, the principles of DCA are introduced and followed by the proposed extended deterministic DCA. Section 4 applies the proposed algorithm for the dynamic JSSP. Finally, the application is tested in Section 5, which is followed by conclusions in Section 6.

2 Dynamic JSSP Definition

The classic JSSP can be described as follows [12, 13]: given n jobs, each consists of m operations to be processed on m machines. Each job should be processed by

each machine only once. The processing time and sequence are known in advance. Each job cannot be operated by more than one machine simultaneously, and each machine can manage at most one operation at a time. The task is to schedule all the operations of jobs on each machine to achieve the predefined performance measurement, such as minimizing the makespan, flowtime or tardiness. Compared with the static JSSP, the dynamic JSSP violates the non-operation-disruption assumption due to the existence of unexpected disruptions. In this paper, we consider five kinds of disturbances that commonly occur in the real world. They are (i) machine breakdown, (ii) new job arrival, (iii) job cancellation, (iv) delay in processing time, and (v) shortened processing time, with the objective of makespan minimization. When the system need rescheduling, we assume that the uncompleted operation being processed by the workable machine at the rescheduling point should not be interrupted and is allowed to finish its current job; though the interrupted operation due to machine breakdown should restart its work after rescheduling, and all jobs are immediately resumed to execution once the rescheduling is completed. Here, the rescheduling process is defined as the complete regeneration of a new scheduling plan for the current unfinished operations, which is equivalent to solving a static JSSP. Other factors, such as machine's setup time, transportation time, immediate storage of jobs, and the rescheduling calculation time are not considered.

3 The Extended Deterministic Dendritic Cell Algorithm

Dendritic Cell Algorithm (DCA), first proposed in 2005 [11], has become a new breed of Artificial Immune Systems (AIS) [14]. It is an immune-inspired algorithm that simulates the dendritic cell's behavior and is developed based on the Danger Theory [15]. The theory behind suggests that the immune system does not discriminate on the basis of self or non-self, but on the balance between the concentration of danger and safe signals within the tissue of the body. DCA has distinct advantages of low processing demand and no training period required. As a result, there are a number of successful adaptations of DCA for intrusion detection, sensor networks, mobile robotics, and port scan detection [16-18].

3.1 The Principles of DCA

DCA simulates the function of dendritic cells (DCs) which are responsible for collecting and combining the signals from the pathogens and host cells in the tissue, and then informing the immune system of any changes in signal concentration. DCs take on three states: the immature (iDC), semi-mature (smDC), and mature (mDC) states. Initially, DCs are immature with different

migration thresholds and responsible for collecting inputs, including multiple antigens and signals. These input signals are often categorized into four types: PAMPs (P), danger signals (D), safe signals (S), and inflammation (I), which represent the pathogenic status, indicate changes in behavior, mean normality, and amplify the effects of other signals respectively. Then, the iDCs quantify and process these signals to obtain the concentration of co-stimulative molecules (CSM), semi-mature cytokines (Semi), and mature cytokines (Mat) according to the signal transformation function of Eq. (1) [17] with the weights, shown in Table 1, which are derived from empirically immunological data [17].

$$C_k = \frac{W_{Pk} \times C_P + W_{Dk} \times C_D + W_{Sk} \times C_S}{|W_{Pk}| + |W_{Dk}| + |W_{Sk}|} \times \frac{1 + C_I}{2} \left(k = 0,1,2\right) \tag{1}$$

Table 1. Weights for Signal Processing

W_{ik}	$i=P$	$i=D$	$i=S$
k=0 (CSM)	2	1	3
k=1 (Semi)	0	0	1
k=2 (Mat)	2	1	-3

Here, C represents the concentration, and W represents weight, with subscript denoting the four signals and three output cytokines (k = 0, 1, 2 is corresponding to CSM, Semi, Mat respectively). Once the CSM of an iDC exceeds its migration threshold, it has to transfer to the mature or semi-mature state, which is determined by the relative concentration of Semi and Mat. That is to say, if the concentration of mature cytokines (Mat) is larger than the semi-mature cytokines (Semi), the iDC will migrate to mDC, otherwise, it turns to smDC. When all DCs migrate, each antigen type is assigned a context called the mature context antigen value (MCAV), which shows its anomaly degree and is calculated as the proportion that an antigen has been presented as mature among all migrated DCs. Finally, by comparing the predefined anomaly threshold with MCAV, the antigen type is identified as normal or anomaly. As such, DCA is a population-based algorithm that converts input data into a form of antigen-plus-context to make binary decisions in an uncertain and dynamic problem space.

3.2 The Extended Deterministic DCA (dDCA)

With regards to the original DCA, Greensmith and Aickelin [19] pointed out that all existed versions of DCA had used a relatively large number of parameters and random elements which made the algorithm open to various criticisms. As a result, they proposed a comparable and controllable form of the DCA, called deterministic DCA (dDCA) [19]. However, this version only considers two signal

categories, the safe and danger signal respectively. Additionally, when computing the concentration for the semi-mature cytokines based on the weights in the original DCA, the weight W_{s1}, unlike other weights, have no effect on Semi concentration C_1. To solve the above problems, in this paper, a new version of DCA, called extended dDCA, is proposed. This algorithm is derived based on dDCA but considers all existed types of signals stated in the original DCA. The concentration of CSM is calculated using Eq. (2), and the output context value of an individual DC is reduced to one factor C_{output}, according to Eq. (3), of which a positive number indicates an anomalous context.

$$C_{csm} = (2 \times C_P + C_D + 3 \times C_S) \times \frac{1 + C_I}{2} \tag{2}$$

$$C_{output} = (2 \times C_P + C_D - 3 \times C_S - C_S) \times \frac{1 + C_I}{2} = (2 \times C_P + C_D - 4C_S) \times \frac{1 + C_I}{2} \tag{3}$$

After all DCs migrate to their terminal states, each antigen type is given a MCAV. As the output concentration is no longer a binary value but having a real-valued context, the original DCA for MCAV calculation is not applicable. The new function for MCAV calculation is given by Eq. (4).

$$MCAV = \sum_{C_{output} > 0} C_{output} \times \frac{N(C_{output} > 0)}{N(DC)} + \sum_{C_{output} < 0} C_{output} \times \frac{N(C_{output} < 0)}{N(DC)} \tag{4}$$

where $N(DC)$, $N(C_{output} > 0)$, and $N(C_{output} < 0)$ represent the number of DCs, mature DCs, immature DCs respectively. We consider the number of different DCs here, as the frequency of an antigen to become anomalous or normal is an important factor that reduces the negative effect on a certain antigen context caused by a certain DC having a dominantly larger output concentration than others, but belongs to a much smaller DC group. This MCAV calculated by Eq. (4) is then used to compare with the anomaly threshold and the extended deterministic DCA outputs its decision variable.

4 Extended dDCA for Dynamic JSSP

In this section, the extended dDCA is applied to solve dynamic JSSP with the objective of makespan minimization to decide when to trigger the rescheduling process when sudden unexpected events occurred. Here, the rescheduling process is equal to solving a static JSSP that all information is known at the point. We can obtain high quality solutions, and optimal solutions in some cases, in reasonable time using the AIS-based hybrid algorithm proposed in [13] which is demonstrated to be applicable and efficient for static JSSP. To each problem, it is possible to obtain many different schedules with the same optimal objective. To increase the adaptability and flexibility of each operation, we consider the slack time of each operation for the candidate scheduling. The schedule having the largest total slack time and the largest number of operations whose slack time is

more than 0 is selected as the best choice. The more slack time exists and the larger the number of operations with slack time, the better is the solution. This can reduce the negative effect in dynamic JSSP caused by the fluctuation of each operation's process time.

In dynamic JSSP, the set of disturbances occurred during the pre-arranged processing system is considered as the antigen. The four kinds of signals are defined as follows:

(1) **PAMPs (P)**: a measure of abnormal behavior when the unexpected events have significant and unacceptable negative effect on the objective if no rescheduling occurs.

(2) **Danger Signal (D)**: a signature that shows the small amount and endurable negative effect caused by the disturbances without rescheduling process.

(3) **Safe Signal (S)**: a measure of normal behavior when the disturbance has no negative effect on the scheduling process even if the rescheduling is not triggered.

(4) **Inflammation (I)**: a sign that indicates the destructive potential of the disruption. For example, machine breakdown is considered to be much worse than the fluctuation of the operation's process time. A higher concentration of inflammation represents more danger the scheduling system may encounter.

When an unexpected event happens, *s%* DCs are picked up from the DC pool which contains a total of *N* DCs. These selected ones are assigned to collect the antigen and signals, and then transform these signals to the output cytokines. In this process, the most important step is to quantify the signals to obtain their concentrations for different kinds of disturbances. In this procedure, two values are compared. One of these is the objective value, namely, makespan M_{old}, before the disturbance happens, while the other is the new generated makespan M_{new} caused by the disturbance without rescheduling process. Here, no rescheduling process means that the job sequence on each machine will not change and each operation will start as early as possible which is also true in the previous scheduling policy. For example, if the machine breakdown event occurs, the current operation on this machine is simply right shifted. By comparing these two items, the signals' concentrations are calculated and categorized by the event type.

Type 1 — Machine Breakdown: The machine *j* fails at time T, and it requires t_j duration for reparation. The interrupted job should be re-processed from the beginning when the machine is available.

Set $\Delta T_j = | T + t_j - T_{j\_earliest} |$, $\Delta M = M_{new} - M_{old}$ $\left(0 \le \Delta M \le \Delta T_j\right)$,

$$T_{j\_earliest} = \begin{cases} \min\limits_{f_{ij} > T} \{s_{ij}\} & \exists i, f_{ij} > T \\ T + \dfrac{3}{2} t_j & \text{else}\left(\text{machine } j \text{ finishes all jobs}\right) \end{cases}.$$

where s_{ij} and f_{ij} represent the start time and finish time of i^{th} job processed on machine *j*. ΔT_j implies the possible delay time on machine *j* caused by breakdown.

As this disturbance may have significant effect to the whole schedule, we set C_I to be 3, and quantify other signals according to ΔM.

If $\Delta M = 0$, then set $C_s = \Delta T_j / 2$, $C_d = C_p = 0$.

If $0 < \Delta M < \Delta T_j / 2$, then set $C_s = 0$, $C_d = \Delta M$, $C_p = \Delta M / 2$.

If $\Delta T_j / 2 \le \Delta M \le \Delta T_j$, then set $C_s = 0$, $C_d = \Delta T_j - \Delta M$, $C_p = \Delta M$.

Type 2 — Processing Time Delay: At time T, the process time of the j^{th} operation of job i is extended by e_{ij}. As this disturbance may have little impact because some operations have slack time, we set C_1 to be 1.

Set $\Delta M = M_{new} - M_{old} \left(0 \le \Delta M \le e_{ij}\right)$.

If $\Delta M = 0$, then set $C_s = e_{ij} / 2$, $C_d = C_p = 0$.

If $0 < \Delta M < e_{ij} / 2$, then set $C_s = 0$, $C_d = \Delta M$, $C_p = \Delta M / 2$.

If $e_{ij} / 2 \le \Delta M \le e_{ij}$, then set $C_s = 0$, $C_d = e_{ij} - \Delta M$, $C_p = \Delta M$.

Type 3 — Shortened Processing Time: At time T, the process time of the j^{th} operation of job i is shortened by l_{ij}. Similarly to the processing time delay, we set C_1 to be 1.

Set $\Delta M = M_{old} - M_{new} \left(0 \le \Delta M \le l_{ij}\right)$.

If $\Delta M = l_{ij}$, then set $C_s = l_{ij} / 2$, $C_d = C_p = 0$.

If $l_{ij} / 2 < \Delta M < l_{ij}$, then set $C_s = 0$, $C_d = l_{ij} - \Delta M$, $C_p = (l_{ij} - \Delta M) / 2$.

If $0 \le \Delta M \le l_{ij} / 2$, then set $C_s = 0$, $C_d = \Delta M$, $C_p = l_{ij} - \Delta M$.

Type 4 — New Job Arrival: A new job is arrived at time T. It has m operations with the processing time p_{n+1j} (j=1,2,…,m). Without the rescheduling process, the new arrived job is scheduled at the end, that is to say, each operation of this job is processed when the required machine finishes processing all previous jobs. This disturbance can be regarded as a destructive event, so set C_1 to be 3.

Set $\Delta M = M_{new} - M_{old} \left(0 \le \Delta M \le \sum_j p_{n+1j}\right)$.

Set $\Delta Inc = \dfrac{\sum_j p_{n+1j}}{\sum_j \sum_{1 \le i \le n} p_{ij}} \times M_{old}$, $\rho = \dfrac{\sum_j p_{n+1j}}{\Delta Inc}$.

where p_{ij} represents the processing time required by the j^{th} operation of job i (i=1,2,…n, j=1,2,…,m). Here, ΔInc means the acceptable effect on makespan.

If $\rho \le 2$ or $0 \le \Delta M \le \Delta Inc$, then set $C_s = \Delta Inc - \Delta M / 2$, $C_d = C_p = 0$.

If $\rho > 2$ and $\Delta Inc < \Delta M < 2\Delta Inc$, then set $C_s = 0, C_d = \Delta M - \Delta Inc, C_p = (\Delta M - \Delta Inc) / 2$.

If $\rho > 2$ and $2\Delta Inc \le \Delta M \le \sum_j p_{n+1j}$, then set $C_s = 0$, $C_d = \sum_j p_{n+1j} - \Delta M$, $C_p = \Delta M - \Delta Inc$.

Type 5 — Job Cancellation: At time T, the job k is cancelled and v operations of job k have already been finished. It has a total of m operations with the processing time p_{kj} (j=1,2,…,v-1,v,v+1,…,m).

Set $\Delta M = M_{old} - M_{new}\left(0 \le \Delta M \le \sum_{v<j\le m} p_{kj}\right)$.

Set $\Delta Dec = \frac{\sum_{v<j\le m} p_{kj}}{\sum_i \sum_j p_{ij}} \times M_{old}\left(0 \le \Delta Dec \le \sum_{v<j\le m} p_{kj}\right)$.

Here, p_{ij} has the same meaning as the definition stated above. As this disruption may or may not be destructive, we set C_I to be 2.

If $\Delta Dec \le \Delta M \le \sum_{v<j\le m} p_{kj}$, then set $C_s = \Delta M / 2$, $C_d = C_p = 0$.

If $\Delta Dec/2 < \Delta M < \Delta Dec$, then set $C_s = 0$, $C_d = \Delta Dec - \Delta M$, $C_p = (\Delta Dec - \Delta M)/2$.

If $0 \le \Delta M \le \Delta Dec/2$, then set $C_s = 0$, $C_d = \Delta M$, $C_p = \Delta Dec - \Delta M$.

After quantifying the signals, the CSM concentration and the output cytokines can be calculated according to Eq. (2) and (3). Then update the accumulative CSM concentration and the output context value. If the accumulative CSM concentration exceeds the predefined migration threshold T_{DC} of each DC, the DC will migrate to its terminal state. If its accumulative C_{output} is below 0, the migrated DC is labeled semi-mature, otherwise, it becomes mature. At the beginning, each DC is randomly assigned a migration threshold in a certain range upon its creation. In this problem, we define this range as $[\alpha \times M_{old} \times (W_{P0}+W_{D0}+W_{S0}), \beta \times M_{old} \times (W_{P0}+W_{D0}+W_{S0})]$ ($\beta > \alpha > 0$) which is based on the acceptable negative effect on the makespan. When the CSM concentration of a DC exceeds its threshold, it shows that the accumulative negative effect on the objective is too large that the DC should stop collecting new items and transfer to the next status for evaluation. If the scheduling system is set to be less sensitive to the disturbances, α and β should be set to larger values; while smaller α and β are more suitable for the system with higher demand.

Once a DC migrates, a new DC will be randomly generated in the DC pool. After all selected DCs finish collecting the signals on this unexpected event, if someone migrates, check whether more than $r\% \times N$ DCs have migrated to their terminal states. In this case, it means that there are enough DCs representing the dynamic information of the system. Hence, it is time to decide whether to trigger the rescheduling process by calculating MCAV according to Eq. (4). If MCAV is larger than 0, the antigen which represent a series of disturbances is regarded as anomaly and the scheduling system should trigger the rescheduling process at this point. Otherwise, the antigen is normal that nothing need to be done to the system. No matter which action is adopted, the antigen and all DCs should be cleared, and a new DC pool will be regenerated randomly for the following disruptions occurred in the system.

The pseudocode of the extended dDCA for dynamic JSSP is given in Figure 1.

```
Initialize: Set Parameters: N, s%, α, β, r%.
        Generate N DCs with the random migration threshold T_DC in the predefined range.
        Generate an antigen without any information.
Begin
  While (a disturbance occurs)
    Add this disturbance to the antigen;
    Quantify the signals and Calculate their concentrations for the disturbance;
    Randomly select s%×N DCs from the DC pool;
    For each selected DCs
      Collect the signals and Calculate C_csm and C_output according to Eq. (2) and (3);
      Update the accumulative C_csm and C_output;
      If (accumulative C_csm > its migration threshold T_DC)
        Remove this DC from the pool and migrate this DC;
        Generate a new DC for the pool to replace the migrated one;
      End If
    End For
    If (the number migrated DCs > r%×N)
      Calculate MCAV according to Eq. (4);
      If (MCAV > 0)
        Trigger the rescheduling process;
      End If          // This is the result of this algorithm
      Clear the information stored in the antigen;
      Clear migrated DCs and the DC pool;
      Create the new DC pool by generating N DCs with the random migration threshold;
    End If
  End While
End
```

Figure 1. Pseudocode of the extended dDCA for Dynamic JSSP

5 Experiments and Results

Classical benchmark problems for static JSSP can be found in the OR-Library [20]. However there is no benchmark problem for this class of dynamic JSSP where only some unexpected disturbances occurred in the pre-arranged scheduling process. As such, we design a number of randomly generated cases for the static benchmark JSSP processing procedure. Each case consists of a series of different types of unexpected events occurred at random time intervals.

When randomly creating the disruptions, the problem's characteristic, especially its size and process time scale, should be considered. For Type 2 and 3

disturbances, the extended/shortened processing time should be reasonable, i.e. no larger than its original processing time ($0<e_{ij}, l_{ij}\leq p_{ij}$). For Type 4 events, the process time that the randomly inserted job works on each machine is generated randomly between the minimum processing time of all existed operations and the maximum time.

To demonstrate the feasibility and efficiency of our algorithm, we compare our algorithm with another two policies for this dynamic JSSP. One is that no rescheduling occurs for all disruptions, which does not change the job sequence on each machine and each operation will start as early as possible. The other is the event-driven rescheduling that the rescheduling is triggered once an unexpected event occurs. Without considering the stability of the scheduling system and neglecting the time for regenerating the new optimal scheduling or reorganizing the ongoing system, it is obviously that the event-driven rescheduling approach can achieve the best scheduling result for the objective, while no rescheduling policy will obtain the worst outcome.

In this experiment, ten problems of four different scales (n jobs × m machines) are selected from the static benchmark JSSP to test the proposed algorithm, including FT06 (6×6), FT10 and ORB01-ORB03 (10×10), LA11-LA14 (20×5), ABZ7 (20×15). The parameters are defined based on the sensitivity of the system and customer demand. In the experiments, they are set as follows: N=100, $s\%$=0.8, α=10%, β=15%, $r\%$=0.7. For each problem except FT06, twenty unexpected events are randomly inserted in its processing procedure. We compare the makespan results, showing in Table 2, among three policies after all disturbances arrive. The last column lists the rescheduling times triggered by the extended dDCA. Furthermore, FT06 (6×6) is chosen as an example to present the detailed experimental results. For FT06, we design four cases which include 3, 5, 8, 10 unexpected events respectively. The results in each step are shown in Tables 3~6, listing the makespan value after dealing with every event by each policy. The symbol "√" means that the rescheduling process is triggered by the proposed algorithm when this event occurs.

Table 2. Results (Makespan Values) of the different policies for Dynamic JSSP

Problem	Best Makespan of Static JSSP	No Rescheduling	Event-driven Rescheduling	Extended dDCA	Rescheduling Times of Extended dDCA
FT10	930	1585	1147	1159	3
ORB01	1059	1979	1344	1372	5
ORB02	888	1830	1025	1072	6
ORB03	1005	1762	1198	1226	4
LA11	1222	1941	1362	1403	5
LA12	1039	2015	1239	1274	6
LA13	1150	1742	1200	1237	4
LA14	1292	1956	1441	1483	4
ABZ7	656	998	711	726	4

Table 3. Results (Makespan Values) of the different policies for Dynamic JSSP of FT06—Case 1

Case 1	No Rescheduling	Event-driven Rescheduling	Extended dDCA
Event 1	67	60	**60 (√)**
Event 2	67	60	60
Event 3	67	60	60

Table 4. Results (Makespan Values) of the different policies for Dynamic JSSP of FT06—Case 2

Case 2	No Rescheduling	Event-driven Rescheduling	Extended dDCA
Event 1	55	55	55
Event 2	81	61	**61 (√)**
Event 3	81	61	61
Event 4	90	61	61
Event 5	90	64	64

Table 5. Results (Makespan Values) of the different policies for Dynamic JSSP of FT06—Case 3

Case 3	No Rescheduling	Event-driven Rescheduling	Extended dDCA
Event 1	55	55	55
Event 2	53	53	53
Event 3	70	60	**60 (√)**
Event 4	70	66	66
Event 5	69	66	66
Event 6	69	66	66
Event 7	81	**72**	**78**
Event 8	81	72	**72(√)**

Table 6. Results (Makespan Values) of the different policies for Dynamic JSSP of FT06—Case 4

Case 4	No Rescheduling	Event-driven Rescheduling	Extended dDCA
Event 1	55	55	55
Event 2	82	63	**63 (√)**
Event 3	82	63	63
Event 4	79	**59**	**63**
Event 5	86	**59**	**63**
Event 6	86	**60**	**63 (safe antigen) (no rescheduling)**
Event 7	93	**61**	**63**
Event 8	93	76	**76 (√)**
Event 9	92	76	76
Event 10	92	76	76

From the tables, we can see that the extended dDCA performs very well compared with other two policies, as it can achieve acceptable makespan value which is a little worse than the best one achieved by the event-driven rescheduling, but only initiates a few rescheduling processes which increases the stability of the job shop. Compared with other heuristic methods which always face the scaling problem, DCA requires little processing and running time to decide whether to trigger the rescheduling process, because it is a population based algorithm without training periods, random searching process or iterations. Therefore, the algorithm can help the job shop response to the disruptions rapidly.

In short, the extended dDCA performs very well on deciding when to trigger the rescheduling process in the dynamic JSSP. Compared with the event-driven and periodic rescheduling policy, it is efficient to decide when to trigger the rescheduling process. It reduces the rescheduling times to mitigate the negative affect on the stability of the scheduling system as much as possible with acceptable objective values, as it only triggers the rescheduling when the system cannot endure the negative impact caused by the cumulative disruptions. As such, the extended dDCA performs much better than both rescheduling policies under consideration.

6 Conclusions

In this paper, we have developed an extended dDCA to determine the optimal triggering of the rescheduling process for solving dynamic JSSP. This algorithm is derived based on dDCA which makes improvement on the original algorithm as it adopts a full set of parameters and random elements. While the original dDCA considers only two types of signals, the extended dDCA includes all relevant signal types and modifies the MCAV calculation by considering the magnitude of the output value. This algorithm is implemented and successfully applied to the dynamic JSSP, and experimented with ten typical problems of different scales. In the experiment, the algorithm is compared with conventional scheduling algorithms, namely, no rescheduling policy and event-driven rescheduling policy. The results show that the proposed algorithm is competitive as it can reduce the rescheduling times to keep the stability of the ongoing scheduling system and ensure the objective acceptable which is only a little worse than the best one. In addition, the implementation of the algorithm is computational efficient as it consumes little computation time to decide the rescheduling action.

As dynamic job shop scheduling environment is extremely complex, different kinds of disturbances may arise in a different way. Currently, this research focuses on the unexpected disturbances occurred occasionally in the scheduling system. Hence future research will focus on extending the algorithm for more complicated dynamic environment and dealing with the stochastic job shop problems with respect to continuous job arrival.

References

1. Rangsaritratsamee, R., Ferrell, J.W.G., Kurz, M.B.: Dynamic rescheduling that simultaneously considers efficiency and stability. Computers & Industrial Engineering. Vol. 46, pp. 1-15 (2004).
2. Jain, A., Meeran, S.: A state-of-the-art review of job-shop scheduling techniques. European Journal of Operations Research. Vol. 113, pp. 390-434 (1999).
3. Vinod, V., Sridharan, R.: Dynamic job-shop scheduling with sequence-dependent setup times: simulation modeling and analysis. International Journal of Advanced Manufacturing Technology. Vol. 36, pp. 355-372 (2008).
4. Xiang, W., Lee, H.P.: Ant colony intelligence in multi-agent dynamic manufacturing scheduling. Engineering Applications of Artificial Intelligence. Vol. 21, pp. 73-85 (2008).
5. Subramaniam, V., Ramesh, T., Lee, G.K., Wong, Y.S., Hong, G.S.: Job shop scheduling with dynamic fuzzy selection of dispatching rules. International Journal of Advanced Manufacturing Technology. Vol. 16, pp. 759-764 (2000).
6. Blackstone, J.H., Phillips, D.T., Hogg, G.L.: A state-of-the-art survey of dispatching rules for manufacturing job shop operations. International Journal of Production Research. Vol. 20, pp. 27-45 (1982).
7. Kang, S.G.: Multi-agent based beam search for intelligent production planning and scheduling. PhD Thesis, Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, Hong Kong (2007).
8. Jang, W.S.: Dynamic scheduling of stochastic jobs on a single machine. European Journal of Operational Research. Vol. 138, pp. 518-530 (2002).
9. Sabuncuoglu, I., Bayiz M.: Analysis of reactive scheduling problems in a job shop environment. European Journal of Operational Research. Vol. 126, pp. 567-586 (2000).
10. De Castro, L.N., Timmis, J.: Artificial Immune Systems: A new computational intelligence approach. Springer, New York (2002).
11. Greensmith, J., Aicklin, W., Cayzer, S.: Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. 4th International Conference on Artificial Immune Systems. Vol. 3627, pp. 153-167 (2005).
12. Mascis, A., Pacciarelli, D.: Job-shop scheduling with blocking and no-wait constraints. European Journal of Operational Research. Vol. 143, pp. 498-517 (2002).
13. Qiu, X.N., Lau, H.Y.K.: An AIS-based hybrid algorithm with PSO for job shop scheduling problem. 10th IFCA Workshop on Intelligent Manufacturing Systems. pp. 371-376 (2010).
14. Garrett, S.M.: How do we evaluate artificial immune systems? Evolutionary Computation. Vol. 13, pp. 145-177 (2005).
15. Aickelin, U., Bentley, P., Cayzer, S., Kim, J., McLeod, J.: Danger theory: The link between AIS and IDS? 2th International Conference on Artificial Immune Systems. Vol. 2787, pp. 147-155 (2003).
16. Al-Hammadi, Y., Aickelin, U., Greensmith, J.: DCA for Bot Detection. 2008 IEEE World Congress on Computational Intelligence. pp. 1807-1816 (2008).
17. Greensmith, J.: The dendritic cell algorithm. PhD Thesis, School of Computer Science, University of Nottingham, UK (2007).
18. Li, X., Fu, H.D., Huang, S.L.: Design of a dendritic cells inspired model based on danger theory for intrusion detection system. Proceedings of 2008 IEEE International Conference on Networking, Sensing and Control. Vol. 2, pp. 1137-1141 (2008).
19. Greensmith, J., Aickelin, U.: The deterministic dendritic cell algorithm. 7th International Conference on Artificial Immune Systems. Vol. 5132, pp. 291-302 (2008).
20. Beasley, J.: OR-Library: Distributing test problems by electronic mail. The Journal of the Operational Research Society. Vol. 41, pp. 1069-1072 (1990).

Reinforcement Learning for Scheduling of Maintenance

Michael Knowles, David Baglee[1] and Stefan Wermter[2]

Abstract Improving maintenance scheduling has become an area of crucial importance in recent years. Condition-based maintenance (CBM) has started to move away from scheduled maintenance by providing an indication of the likelihood of failure. Improving the timing of maintenance based on this information to maintain high reliability without resorting to over-maintenance remains, however, a problem. In this paper we propose Reinforcement Learning (RL), to improve long term reward for a multistage decision based on feedback given either during or at the end of a sequence of actions, as a potential solution to this problem. Several indicative scenarios are presented and simulated experiments illustrate the performance of RL in this application.

1 Introduction

Condition-based maintenance (CBM) is an area which has received substantial attention in recent years. Prior to the advent of CBM, maintenance was either reactive, repairing faults as they occurred which led to downtime and the potential for extended damage due to failed or failing parts, or planned preventative maintenance which sought to prevent failures by performing maintenance on a pre-planned fixed schedule, where the reliability and efficiency of this approach depended on the appropriateness of the schedule [1,2].

CBM involves performing some measurement of the condition of equipment so as to infer the maintenance needs. Condition data is generally compiled from sensors recording various aspects of the equipment's condition, including vibration measurements, temperature, fluid pressure and lubricant condition. Typically a series of thresholds are defined which trigger an intervention when the measurements go above these thresholds [3, 4]. Furthermore, several levels of

[1] Institute for Automotive and Manufacturing Advanced Practice (AMAP), University of Sunderland, Colima Avenue, Sunderland, SR5 3XB, UK

[2] Knowledge Technology Group, Department of Informatics, University of Hamburg, Vogt Koelln Str. 30, 22527 Hamburg, Germany

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_31,

alert are set depending on the level of seriousness of the fault. To fully exploit condition measurements, it is, however, necessary to be able to predict the precise implications of a given action under a particular set of condition measurements. This can be achieved using combinational limits which trigger alerts when several thresholds are passed but these must be set up either empirically or through detailed analysis if they are to optimise reliability and efficiency [5]. Under-maintenance due to optimistic threshold setting will lead to failures while over-maintenance will lead to inefficiency as maintenance is performed too frequently.

An increasingly important factor in maintenance scheduling is energy efficiency [6,7,8,9,10]. Many types of equipment become inefficient if they are not correctly maintained. This can lead to a complex set of criteria for the optimisation of maintenance. Factors which can influence the optimisation include reliability targets, failure penalties, downtime costs, preventative maintenance costs and energy consumption/efficiency. A further complication is that the rate at which maintenance becomes necessary is often partially determined by usage and as such this can vary based on the activities of the organisation in question. Therefore, optimising maintenance schedules can be a highly complex activity. Since this activity is essentially a long-term optimisation over a series of short term decisions, it is our hypothesis that reinforcement learning (RL) is well suited to this task.

Due to the use of a simple, final reward, reinforcement learning has found applications in interaction scenarios where an agent receives feedback from a user at the end of a sequence of actions such as dialogue management [11], visual homing and navigation [12,13,14,15,16], human-computer/robot interaction [17], robot navigation [18,19] and for learning skills in the Robocup Soccer Competition [20,21,22]. There have already been some initial attempts to explore reinforcement learning for restricted tasks in scheduling, routing, and network optimisation. [23,24,25,26,27,28,29,30,31,32] Our approach differs from these since it offers a practical application for RL in a real-world online environment. In this application RL will not only adapt to the broad properties of the problem but also to the individual properties of the equipment used. RL is outlined in the subsequent section and the remainder of the paper is devoted to demonstrative simulations involving the use of RL to schedule maintenance. The paper is concluded with discussions of the results and suggestions for future work.

2 Reinforcement Learning

Reinforcement learning is a machine learning paradigm based on the psychological concept of reinforcement, where the likelihood of a particular behaviour is increased by offering some reward when the behaviour occurs. In computational terms RL is concerned with maximising long term reward following a sequence of actions [33,34,35,36,37]. Many RL algorithms have been

proposed [37] including Q-Learning [38]. SARSA [39], Temporal Distance Learning [40] and actor-critic learning [41]. The experiments presented here have used the Q-Learning algorithm first proposed by Watkins [38]. Q-Learning was selected due to the simplicity of its formulation, the ease with which parameters can be adjusted and empirical evidence of faster convergence than some other techniques [36]. Q-Learning is based on learning the expected reward, *Q,* achieved when a particular action, *a*, is undertaken when in a particular state, *s,* given that a policy, π, is followed thereafter:

$$Q(s,a) = E\left[R|s,\pi,a\right] \tag{1}$$

The Q-Values are updated with the following equation at each epoch:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left(r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right) \tag{2}$$

where *r* is the reward, α is the learning rate and γ represents the discount factor applied to future rewards. Adjusting the value of γ regulates the influence of future reward on the current decision, i.e. it controls how forward-looking the system is in seeking to maximise future reward. A key component of RL is the balance between exploration and known reward. In the maintenance scenario this would occur if the agent learned that performing maintenance at every time step produces a known reward causing it to never learn that a greater reward may be possible by taking a different policy. This scenario is avoided by using the Q-values to bias the action selection rather than providing a definitive choice. Another key aspect of reinforcement learning systems is ensuring convergence. Convergence can be ensured if α takes successively decreasing values subject to certain constraints [42]. Based on the above formulation and properties of the Q-Learning algorithm a series of experiments can now be performed.

3 Problem Formulation

In order to test the suitability of RL to the maintenance scheduling problem, it is necessary to define some indicative scenarios which can form the basis of simulated experiments. These simulations will involve two interacting components, a plant-model and a reinforcement learning model. The plant model provides the RL module with an indication of a current condition, the RL module then decides whether to execute a particular maintenance task. This is similar to the optimal control scenario described by Sutton, Barto and Williams [36] and is

illustrated in figure 1 below. If maintenance is not performed then a failure may or may not occur. If the plant does not fail then a profit is returned as a reward. If the system does fail then a repair cost is deducted from the profit. If the RL module decides to perform maintenance then the system will not fail but a maintenance cost is deducted from the profit. The maintenance cost is considerably lower than the failure cost as is typical in real world scenarios. Thus at each time step the RL module must decide between a known, moderate reward by performing maintenance or risking no maintenance which could incur either a high reward in the event of no failure or a low reward if the plant fails

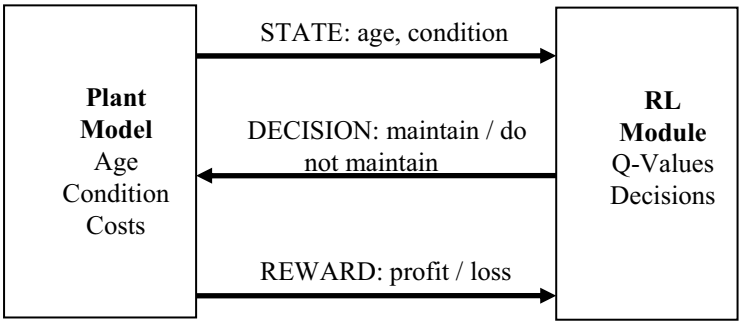


Fig. 1. Simulation Architecture

3.1 Plant model

The objective is to maximise reliability, i.e. the rate at which the equipment in question suffers a failure. In mathematical terms, the reliability function *R(t)* represents the likelihood that a system will run for a given time *t* without failure:

$$R(t) = P(T > t) \tag{3}$$

where *T* is the failure time.

In the experiments described below, the plant model consists of a reliability function which is based on various combinations of variables including:

- Time since last maintenance, *t.* It is assumed in all cases that the likelihood of a failure increases with *t.*

- Condition, *c,* which represents the condition of the plant, independent of the time since the last maintenance. After maintenance the value of condition is set to 1, and will decrease by a random amount after each time step. The likelihood of failure is inversely proportional to the value of *c.*

For implementation purposes the reliability function is formulated in terms of the failure probability which is a function of the above variables, and represents the probability that a failure will occur for a given state *(t,c)*. Several failure probability functions are used in the following experiments to illustrate various levels of complexity. These functions are given in the following section. Once the decision whether to maintain has been taken, the plant model will calculate the reward as described above based on the profit, repair cost and maintenance cost. In some cases the profit will also reduce at each time step to simulate the effects of increasing running costs (i.e. due to increased energy consumption etc) due to deteriorating condition. Once again various functions are used to illustrate different types of system, the functions are given for each experiment in the following section.

3.2 Reinforcement Learning Model

In order to develop a maintenance model based on Q-Learning it is necessary to define the system state, the available actions and the Q function. The objective is to present the system with a stimulus and ask it a question, before providing reward based on the answer. In the experiments performed, the stimulus will be a set of state variables from the plant model which will consist of time since last maintenance, *t*, and condition, *c*. The response will be a decision to perform maintenance or not based on these state variables alone. This decision will be biased by the Q-Values for the two actions. Thus even if there is a larger expected reward, represented by a larger Q-value, available for a given action it is still possible for the other action to be taken in order to gain an opportunity to explore new actions. .Once the maintenance decision has been passed back to the plant model, the RL module will receive its reward. Based upon this reward the Q value for the selected action in the given state is updated according to equation 4:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right) \qquad (4)$$

It should be emphasised that the RL module only sees the state variables and reward which in a real-world application are measurable. The RL module has no knowledge of the reliability function or reward functions of the plant model.

Actions are selected with the probabilities of maintenance actions being in direct proportion to the relative Q-Values. In order to ensure convergence, it is necessary for the value of α to decrease through the course of the trials subject to certain conditions [42,43]. This is achieved using the typical scheme:

$$\alpha(s,a) = \frac{1}{n(s,a)} \tag{5}$$

Where *n(s,a)* represents the number of times *Q(s,a)* is visited.

3.3 Experiments

In order to examine the performance of the RL algorithm in the maintenance scheduling scenario, four simulated experiments were performed using Matlab and these are described below. The first scenario presented is the most basic with the level of complexity increasing thereafter. In order to quantify the performance of the reinforcement learning system two metrics are used. The expected reward is calculated by running in validation mode 10000 times between each training iteration of the learning algorithm and averaging the reward accrued. Validation mode involves using the current policy to operate the plant starting from $t = 0$. Since the purpose of these tests is to measure the performance of a particular policy, there is no explorative behaviour in validation mode, i.e. the action with the highest Q-Value in a given state will always be selected. There is no learning or update of the Q-Value in validation mode. The other metric used is the Mean Time Between Failures (MTBF) which is a commonly used reliability metric. There are various formulations of MTBF, in this instance it represents the mean number of epochs between each occasion the system fails in validation mode.

4 Results

4.1 Level 1: Basic Model

Here a simple system involving a running cost and a failure/repair cost is simulated. While this system is simplified it serves as an effective demonstrator of the application and as an introduction to the more elaborate, realistic scenarios below. The details are as follows. The system is capable of making a profit of 100

units at each epoch. The system has a failure probability of 0 which increases linearly by 0.05 each epoch as described in equation 6:

$$p_{fail}(t) = 0.05t \tag{6}$$

The reward available at each epoch is given by:

$$r_t = \begin{cases} 100 & \text{no maintenance or failure} \\ 100 - c_r & \text{no maintenance performed, system fails} \\ 100 - c_m & \text{maintenance performed} \end{cases} \tag{7}$$

where c_r = 120 represents the repair cost when the system fails and c_m = 30 is the maintenance cost. The system is simulated for 1000 epochs. In this instance, the decision as to whether or not to perform maintenance is taken randomly for training purposes. The system was tested with the reward discount factor γ set to 0.1. This value was determined empirically and found to be successful. The resulting Q-Values are shown in figure 2. It can be seen that maintenance becomes a more favorable option after 4 epochs. This is significant since the expected rewards for the two actions, calculated statistically using equations 6 and 7, are equal at 5 epochs with maintenance having a higher value than no maintenance before 5 epochs and a lower value after, as shown in table 1. Figure 3 shows the expected reward which can be seen to quickly converge, and the MTBF. It can be seen that the dominant MTBF is not the optimal value achieved. This is due to the agent attempting to achieve optimal long-term reward by delaying maintenance as long as it considers prudent.

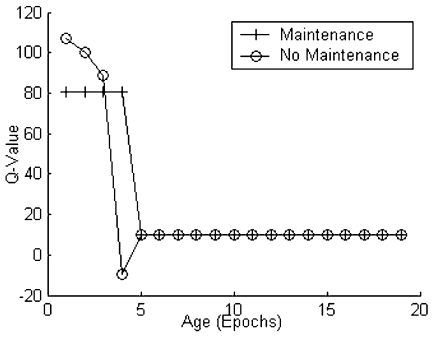


Fig. 2. Q-Values for Level 1

Table 1. Expected Rewards

T	$p_{fail}(t)$	E(r_t|maintenance)	E(r_t|no maintenance)
1	0.05	70	94
2	0.1	70	88
3	0.15	70	82
4	0.2	70	76
5	0.25	70	70
6	0.3	70	64
7	0.35	70	58

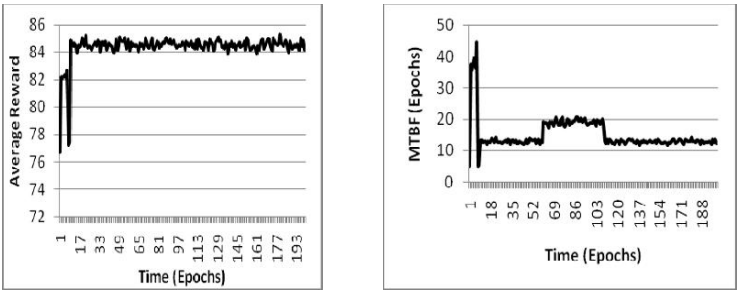


Fig. 3. Average Reward and MTBF for Level 1

4.2 Level 2: Condition Data

Here we provide the system with a measure of its current condition. The failure probability function is now modified to involve the condition variable c as discussed above and is shown in equation 8.

$$p_{fail}\left(t, c(t)\right) = \max\left(0.2 + 0.05t, 1 - c(t)\right) \tag{8}$$

The value of condition is updated at each time step as described in equation 9.

$$c(t) = c(t-1) - 0.1rand \tag{9}$$

Where *rand* represents a uniformly distributed random number in the range 0-1. The reward function remains as specified in equation 7. The results of the simulation can be seen in figure 4. It can be seen that the algorithm successfully converges on a policy yielding an average reward in the region of 81 units. Again, the final value of MTBF is suboptimal, however the optimal value corresponds with a lower level of reward which is the criteria against which the algorithm is optimising.

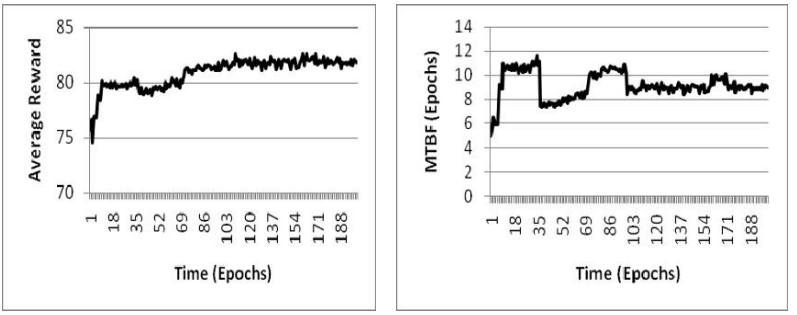


Fig. 4. Average Reward for Level 2.

4.3 Level 3: Energy Consumption Data

This scenario involves the simple reliability function from Level 1 as described by equation 6. Here, however, the running costs of the system increase at each time step to simulate an increase in energy usage due to a deteriorating condition. This is distinct from the above condition scenario where the running costs are not directly influenced until the equipment fails. Thus the profit available at each epoch reduces by 5 units at each time step after maintenance as described by equation 10.

$$r_t = \begin{cases} 100 - 5t & \text{no maintenance or failure} \\ 100 - 5t - c_r & \text{no maintenance performed, system fails} \\ 100 - c_m & \text{maintenance performed} \end{cases} \tag{10}$$

The Q-values are shown in figure 5, average reward and MTBF in figure 6. It can be seen that the average reward converges, but on occasion loses its optimality temporarily. This appears to occur in the unlikely event of multiple successive failures in the learning algorithm but is rapidly corrected. The previously observed phenomenon regarding the sub-optimal MTBF is clearly illustrated here as the

MTBF rises during periods where the policy becomes sub-optimal in terms of reward.

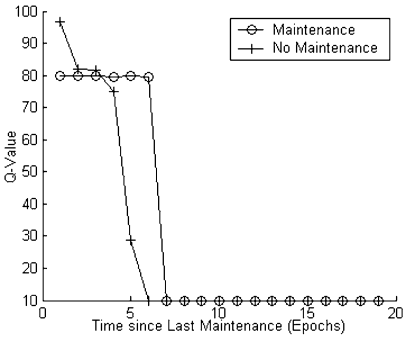


Fig. 5. Q-Values for Level 3.

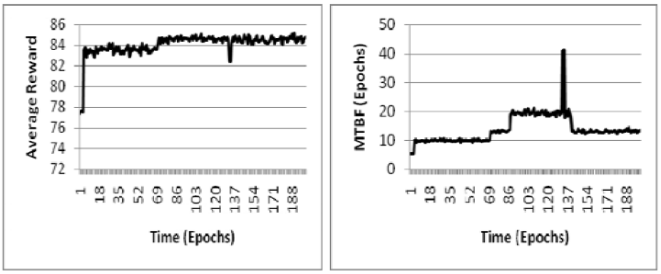


Fig. 6. Average Reward for Level 3.

It should be noted that the effect of a deteriorating condition does not necessarily need to be formulated in terms of direct running costs. The reward offered could be formulated in terms of emissions, cost or other requirements scaled with suitable coefficients to give priority as chosen by the user.

4.4 Level 4: Complex System

In this scenario we combine the above concepts of time since last maintenance, condition measurement and energy usage. Thus the reliability function from Level 2 (equation 8) is used in conjunction with the reward function from level 3 (equation 10). The average reward and MTBF for level 4 are shown in figure 7. As with the previous examples, it can be seen that convergence is achieved.

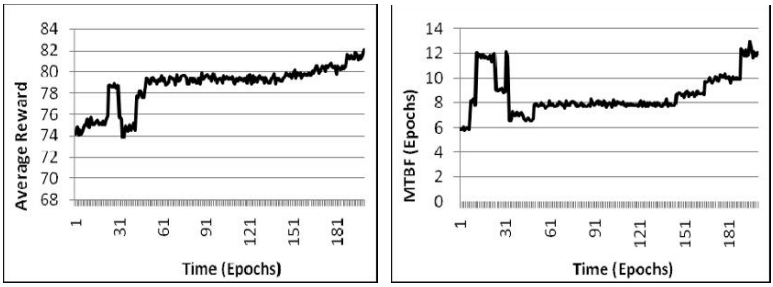


Fig. 7. Average Reward for Level 4.

5 Discussion and Conclusions

A number of benefits of RL have been demonstrated in limited yet realistic scenarios. The approach described has a number of merits including no requirement for any form of internal model and an ability to optimize against a number of criteria and could be applied successfully in a larger maintenance management application. The state described here comprises of the time since last maintenance and simple condition measurements, however the two variables used in the state vector cover the most important factors in a system's reliability and potential improvements to the model would only improve the level of detail represented. The state-space could, for example, be expanded to include factors such as indicators of individual component condition, overall age, more detailed service history etc. As the state space becomes larger, maintaining an estimate of each and every possible Q-Value becomes problematic for scaling the problem size. This can be mitigated by modeling the Q-Function using a function approximator such as a neural network. This is an approach which has been successfully applied in many applications [12,13,44]. The repertoire of actions could be increased to consider different levels of maintenance, each with different availabilities.

Future work in this area will need to probe these questions and address issues including the reliability of such a system in terms of the stability of the Q-Values, the effect of varying the future discount parameter to regulate how long-term the systems decision criteria is and the successful integration of cost based rewards with other parameters against which maintenance should be optimised such as MTBF. Furthermore the needs of industry in developing this application into a useful tool need to be considered to ensure it remains relevant. Issues such as formulating and observing the inner state of the system and the implications of the actual Q-Values in terms of metrics used by maintenance managers such as Return on Investment (ROI) will need to be addressed.

References

1. Grall A., Berenguer C., Dieulle L.: A condition-based maintenance policy for stochastically deteriorating systems. Reliability Engineering & System Safety, Volume 76, Issue 2, Pages 167-180, ISSN 0951-8320, DOI: 10.1016/S0951-8320(01)00148-X.(2002)
2. Bengtsson M.: Standardization Issues in Condition Based Maintenance. In Condition Monitoring and Diagnostic Engineering Management - Proceedings of the 16th International Congress, August 27-29, 2003, Växjö University, Sweden, Edited by Shrivastav, O. and Al-Najjar, B., Växjö University Press, ISBN 91-7636-376-7. (2003)
3. Davies A. (Ed): Handbook of Condition Monitoring - Techniques and Methodology. Springer, 1998 978-0-412-61320-3.(1997)
4. Barron R. (Ed): Engineering Condition Monitoring: Practice, Methods and Applications. Longman, 1996, 978-0582246560.(1996)
5. Wang W.: A model to determine the optimal critical level and the monitoring intervals in condition-based maintenance. International Journal of Production Research, volume 38 No 6 pp 1425 – 1436. (2000)
6. Meier A.: Is that old refrigerator worth saving? Home Energy Magazine http://homeenergy.org/archive/hem.dis.anl.gov/eehem/93/930107.html(1993)
7. Litt B., Megowen A. and Meier A.: Maintenance doesn't necessarily lower energy use. Home Energy Magazine http://homeenergy.org/archive/hem.dis.anl.gov/eehem/93/930108.html. (1993)
8. Techato K-A, Watts D.J. and Chaiprapat S.: Life cycle analysis of retrofitting with high energy efficiency air-conditioner and fluorescent lamp in existing buildings. Energy Policy, Vol. 37, pp 318 – 325. (2009)
9. Boardman B., Lane K., Hinnells M., Banks N., Milne G., Goodwin A. and Fawcett T.: Transforming the UK Cold Market Domestic Equipment and Carbon Dioxide Emissions (DECADE) Report. (1997)
10. Knowles M.J. and Baglee D.:The Role of Maintenance in Energy Saving, 19th MIRCE International Symposium on Engineering and Managing Sustainability - A Reliability, Maintainability and Supportability Perspective, (2009)
11. Singh, S. Litman, D., Kearns M ., and Walker,M. Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System. In Journal of Artificial Intelligence Research (JAIR),Volume 16, pp. 105-133. (2002)
12. Altahhan A., Burn K. Wermter S.: Visual Robot Homing using Sarsa(λ), Whole Image Measure, and Radial Basis Function. Proceedings IEEE IJCNN (2008)
13. Altahhan A.: Conjugate Temporal Difference Methods For Visual Robot Homing. PhD Thesis,University of Sunderland. (2009)
14. Lazaric, A., M. Restelli, Bonarini A.: Reinforcement Learning in Continuous Action Spaces through Sequential Monte Carlo Methods. Twenty First Annual Conference on Neural Information Processing Systems – NIPS. (2007)
15. Sheynikhovich, D., Chavarriaga R., Strosslin T. and Gerstner W.: Spatial Representation and Navigation in a Bio-inspired Robot. Biomimetic Neural Learning for Intelligent Robots. S. Wermter, M.Elshaw and G.Palm, Springer: 245-265. (2005)
16. Asadpour, M. and Siegwart, R.: Compact Q-learning optimized for micro-robots with processing and memory constrains. Robotics and Autonomous Systems, Science Direct, Elsevier. (2004)
17. Knowles, M.J. and Wermter, S.: The Hybrid Integration of Perceptual Symbol Systems and Interactive Reinforcement Learning. 8th International Conference on Hybrid Intelligent Systems. Barcelona, Spain, September 10-12th, (2008)
18. Muse, D. and Wermter, S.: Actor-Critic Learning for Platform-Independent Robot Navigation. Cognitive Computation, Volume 1, Springer New York, pp. 203-220, (2009)

19. Weber, C., Elshaw, M., Wermter, S., Triesch J. and Willmot, C.: Reinforcement Learning Embedded in Brains and Robots, In: Weber, C., Elshaw M., and Mayer N. M. (Eds.) Reinforcement Learning: Theory and Applications. pp. 119-142, I-Tech Education and Publishing, Vienna, Austria. (2008)
20. Stone, P., Sutton R. S. and Kuhlmann G.: Reinforcement learning for robocup soccer keepaway. International Society for Adaptive Behavior 13(3): 165–188 (2005)
21. Taylor M.E. and Stone P.: Towards reinforcement learning representation transfer. In The Autonomous Agents and Multi-Agent Systems Conference (AAMAS-07), Honolulu, Hawaii. (2007)
22. Kalyanakrishnan S., Liu Y. and Stone P.: Half Field Offense in RoboCup Soccer: A Multiagent Reinforcement Learning Case Study. Lecture Notes In Computer Science, Springer (2007)
23. Lokuge, P. and Alahakoon, D.: Reinforcement learning in neuro BDI agents for achieving agent's intentions in vessel berthing applications 19th International Conference on Advanced Information Networking and Applications, 2005. AINA 2005. Volume: 1 Digital Object Identifier: 10.1109/AINA.2005.293, Page(s): 681 - 686 vol.1(2005)
24. Cong Shi, Shicong Meng, Yuanjie Liu, Dingyi Han and Yong Yu: Reinforcement Learning for Query-Oriented Routing Indices in Unstructured Peer-to-Peer Networks, Sixth IEEE International Conference on Peer-to-Peer Computing P2P 2006, Digital Object Identifier: 10.1109/P2P.2006.30, Page(s): 267 - 274 (2006)
25. Cong Shi, Shicong Meng, Yuanjie Liu, Dingyi Han and Yong Yu: Reinforcement Learning for Query-Oriented Routing Indices in Unstructured Peer-to-Peer Networks, Sixth IEEE International Conference on Peer-to-Peer Computing, 2006. P2P 2006.Digital Object Identifier: 10.1109/P2P.2006, Page(s): 267 - 274 (2006).
26 Mattila, V.: Flight time allocation for a fleet of aircraft through reinforcement learning. Simulation Conference, 2007 Winter, Digital Object Identifier: 10.1109/WSC.2007.4419888 Page(s): 2373 - 2373 (2007)
27. Zhang, Y. and Fromherz, M.: Constrained flooding: a robust and efficient routing framework for wireless sensor networks, 20th International Conference on Advanced Information Networking and Applications, 2006. AINA 2006.Volume: 1 Digital Object Identifier: 10.1109/AINA.2006.132 (2006)
28. Chasparis, G.C. and Shamma, J.S.: Efficient network formation by distributed reinforcement 47th IEEE Conference on Decision and Control, 2008. CDC 2008. Digital Object Identifier: 10.1109/CDC.2008.4739163, Page(s): 1690 - 1695 (2008).
29. Usynin, A., Hines, J.W. and Urmanov, A.: Prognostics-Driven Optimal Control for Equipment Performing in Uncertain Environment Aerospace Conference, 2008 IEEE Digital Object Identifier: 10.1109/AERO.2008.4526626, Page(s): 1 – 9 (2008)
30. Lihu, A.and Holban, S.: Top five most promising algorithms in scheduling. 5th International Symposium on Applied Computational Intelligence and Informatics, 2009. SACI '09. Digital Object Identifier: 10.1109/SACI.2009.5136281, Page(s): 397 - 404 (2009).
31. Zhang Huiliang and Huang Shell Ying: BDIE architecture for rational agents.. International Conference on Integration of Knowledge Intensive Multi-Agent Systems, Page(s): 623 - 628 (2005)
32. Malhotra, R., Blasch, E.P. and Johnson, J.D.: Learning sensor-detection policies ., Proceedings of the IEEE 1997 National Aerospace and Electronics Conference, 1997. NAECON 1997Volume: 2 Digital Object Identifier: 10.1109/NAECON.1997.622727 , Page(s): 769 - 776 vol.2 (1997)
33. Sutton, R.S. and Barto, A.G.: Reinforcement Learning: An Introduction, IEEE Transactions on Neural Networks Volume: 9 , Issue: 5 Digital Object Identifier: 10.1109/TNN.1998.712192, Page(s): 1054 - 1054 (1998)
34. Barto, A.G.: Reinforcement learning in the real world 2004. Proceedings. 2004 IEEE International Joint Conference on Neural Networks, Volume: 3 (2004)

35. Barto, A.G. and Dietterich, T.G.: Reinforcement Learning and Its Relationship to Supervised Learning In Si, J., Barto, A.G., Powell, W.B., and Wunsch, D., editors, Handbook of Learning and Approximate Dynamic Programming, pages 47 - 64. Wiley-IEEE Press, (2004)
36. Sutton, R.S., Barto, A.G.: and Williams, R.J.: Reinforcement learning is direct adaptive optimal control Control Systems Magazine, IEEE Volume: 12 , Issue: 2 Digital Object Identifier: 10.1109/37.126844 Publication Year: 1992 , Page(s): 19 - 22
37. Kaebling, L.P., Littman, M.L. and Moore A.W.: Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research, Vol 4, pp 237 – 285. (1996)
38. Watkins, C.J.C.H.: Learning from Delayed Rewards. PhD thesis, Cambridge University, Cambridge, England. (1989).
39. Rummery G.A and Niranjan M.: On-line Q-Learning using connectionist Systems. Technical Report CUED/F-INFENG/TR166, Cambridge University. (1994)
40. Sutton, R.: Learning to predict by the methods of temporal differences. Machine Learning 3 (1),pp 9–44. doi:10.1007/BF00115009. (1988)
41. Foster D.J., Morris, R.G.N.and Dayan, P.: A model of hippocampally dependent navigation, using the temporal learning rule. Hippocampus, Vol. 10, pp. 1-16, (2000)
42. Humphrys, M.: Action Selection methods using Reinforcement Learning , PhD thesis, University of Cambridge, Computer Laboratory (1997)
43. Watkins, C.J.C.H. and Dayan, P.: Technical Note: Q-Learning, Machine Learning 8:279-292. (1992)
44 Sutton R.S.: Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding. Advances in Neural Processing Systems 8, pp1038 – 1044. (1996)

AI IN ACTION

Genetic Evolution and Adaptation of Advanced Protocols for Ad Hoc Network Hardware Systems

Jennifer Jackson[1] and Mark Leeson[2]

Abstract The diversity of future technologies requiring ad hoc networks to operate within unpredicted situations will mean an increase in the required flexibility of the actual protocols used for communicating information. A methodology is proposed to genetically evolve the optimum ad hoc network communication protocol under any given network scenario. The methodology creates and dynamically adapts the communication protocol based upon an alphabet of characteristics and performance metrics using simple protocol mapping techniques and minimisation of a fitness function via a genetic selection process. A scenario has been created to evaluate the performance of the methodology in finding the optimum solution. Preliminary results show that the methodology is able to find the global optimum within several runs. The methodology could be enhanced using Field Programmable Gate Array (FPGA) hardware nodes for real time performance and distributed control.

1 Introduction

A predicted explosion in the demand for mobile services will mean that ad hoc networks of the future must have the ability to interconnect diverse technologies such as wearable computers, and home robots, as well as accommodating environmental conditions that were not premeditated, such as malicious security attacks, failures within the network and sudden changes in topology.

An ad hoc network is characterised by a number of devices, often mobile, connected in an arbitrary manner to form a network without a central controller. Their development began in the 1970s with the appearance of static wireless networks, but they were increasingly adapted, particularly during the 1990s to enable wireless mobility [1]. Today, a number of wireless protocols are in commercial use, but despite this nearly forty year development there are still

[1] Complexity Science, University of Warwick, UK
[2] Engineering, University of Warwick, UK

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_32, © Springer-Verlag London Limited 2011

challenges facing ad hoc communication protocol design. Current protocols are fixed for a given application, but ad hoc networks need to encompass a growing list of requirements that cannot be satisfied by a single fixed protocol. There is therefore a need for network adaptability to cope with the environment and application by choosing the optimum protocol for the given situation. This work exploits the powerful search capabilities of the genetic algorithm, together with simple mapping techniques to evolve optimum protocol designs for a given scenario.

The remainder of this paper is organised as follows: section 2 begins with some background regarding communication protocols, and highlights relevant work. Section 3 details the proposed methodology including the characteristic alphabet, the protocol mapping technique, and the genetic selection process. Section 4 describes the scenario used to test the methodology and section 5 presents the results. Section 6 gives the conclusions of the work followed by acknowledgements and references.

2 Background

2.1 Protocol stack

Communicating from one device to another in an ad hoc network involves a number of layers of interacting processes, from the physical medium such as radio waves to the user software such as a web page. These combined layers form the protocol stack, commonly analysed using the Open Systems Interconnection (OSI) model as shown in Fig.1.

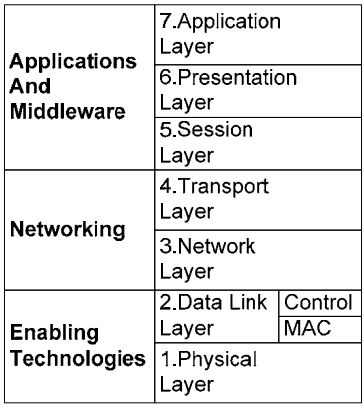


Fig. 1. OSI protocol stack model

Each of the seven layers can contain one or more different sub-protocols. There are many wireless protocol stacks, often only defined for the physical and data link layers because it is these two layers that are mostly concerned with, and affected by, the transmission medium used. In ad hoc networks particular attention needs to be given to the network layer and how the data will be routed due to the constantly changing nature of the ad hoc topology which is not present in other types of networks. Above these layers, where the transmission medium used is of no concern to the application, it is advantageous to share a common language such as Transmission Control Protocol (TCP) when bridging across wired and wireless networks to access information from the internet.

2.2 Related Research

Related research focuses on automated protocol design. Ocenasek and Sveda [2] propose the use of genetic algorithms to develop security protocols. Xue et. al. [3] apply an artificial immune algorithm to make the design of security protocols more secure and reliable. Perrig and Song [4] use an automated technique for security protocol design involving minimising a cost function based upon a set of requirements. Virtanen et. al. [5] suggest the idea of a programmable processor capable of processing several different protocols. Oberg et. al. [6] use a grammar based specification method for hardware synthesis of data communication protocols. None of these ideas however create a protocol dynamically in real time. They are concerned with developing optimum protocols for a set of pre-generated criteria where the network environment is known. Pavlosoglou et. al. [7] however use Selfridge's Pandemonium concept to dynamically emerge an optimum routing protocol for the security of wireless ad hoc networks. Limitations with this method meant that global solutions were not always found. The methodology proposed within this paper improves upon this by using a genetic algorithm approach which is good at finding global solutions, and additionally focuses on multiple layers of the protocol stack to address the most important constituents of a wireless ad hoc protocol.

3 Protocol Methodology

3.1 General Concept

The general concept of the proposed methodology is the creation and adaptation of a communication protocol for a wireless ad hoc network, where the chosen protocol is based upon feedback of the current network performance. The decision

making process has been made at a global level where there is a centralised controller monitoring the network. This allows a *first step* in the investigation of the concept of dynamically creating a communication protocol.

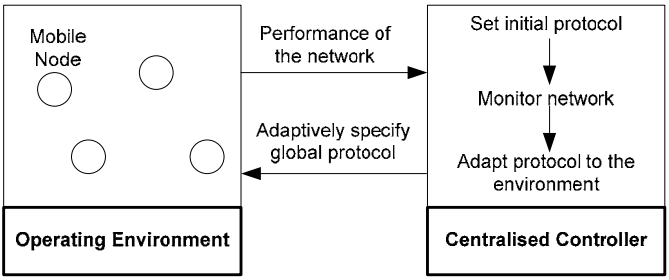


Fig. 2. General concept of the network operating environment

3.2 Alphabet of Characteristics

The functionality of each layer within the protocol stack can be defined by a set of characteristics through classification of all the sub-protocols within it. For example within the physical layer the sub-protocols could be classified according to their transmission frequency or the type of modulation schemes they use. Many such characteristics could be used to classify the sub-protocols, but there is a minimum number needed to uniquely distinguish one sub-protocol from another. This minimum set of characteristics is represented by an alphabet, where each letter of the alphabet represents one particular characteristic. To demonstrate the principle of the methodology three layers of the OSI model have been optimised: a) the physical layer, b) the Media Access Control (MAC) sub-layer, and c) the network layer routing, with the remaining layers fixed. The protocol generation algorithm is used to find the optimum set of characteristic values which map to the optimum protocol.

3.3 Physical Layer Characteristics

The classification of the physical layer sub-protocols available within the simulation tool can be simplified to two independent characteristics as given in Table 1, with each characteristic assigned an alphabet letter. These characteristics allow the solution space to be represented by a two-dimensional vector space as shown in Fig. 5a, where each available protocol for a defined set of internal parameters can be uniquely represented by a point within the vector space. The

range type indicates the kind of network that the wireless protocol was designed for. At one end of the characteristic scale is the Personal Area Network (PAN) designed for the interaction of nodes within close proximity around a person such as communication between a PC and a video camera. In the middle range is the Local Area Network (LAN) designed for interconnecting computers, printers and scanners within office buildings. At the other end of the scale is the Wide Area Network (WAN) designed for connecting devices on a larger scale such as connecting homes and cities to the World Wide Web. The *maximum bit rate* indicates how fast data can be transferred across the network and encompasses the frequency and modulation type of the protocol because at the low end of the characteristic scale low frequencies are used leading to lower bit rates. At the high end of the scale high frequencies are used often with modulation techniques for multiple channels resulting in high bit rates.

Table 1. Physical layer classification; those in italics were not used during simulations

Sub-protocol	A. Range Type	B. Max bit Rate
PHY IEEE802.11a [8]	LAN	High - 54 Mb/s
PHY IEEE802.11b [8]	LAN	Medium - 11Mb/s
PHY IEEE802.16 [9]	*WAN*	*High - 30Mb/s 75Mb/s*
PHY IEEE802.15.4 [10]	*PAN*	*Low - 250kb/s*

3.4 Media Access Control Layer Characteristics

The classification of the MAC layer protocols available within the simulation tool can be represented by three independent characteristics and is given in Table 2. The three characteristics allow representation by a three-dimensional vector space as shown in Fig. 5b. *Contention* is concerned with the ability of the protocol to avoid or resolve collisions when more than one node is attempting to access the channel at the same time. At one end of the characteristic scale are contention-free methods where certain assignments are used to avoid contentions altogether. Contention-based schemes on the other hand are aware of the risk of collisions and take steps to resolve them. Random access methods apply a random wait time if a collision occurs before re-trying, whereas collision resolution or avoidance methods tend to listen to the channel or make an announcement before sending data which subsequently reduces the probability of a collision. *Quality of Service* is a measure of the level of service that data receive when they transfer across the network. The network is expected to guarantee a set of measurable pre-specified service attributes such as end-to-end delay, available bandwidth, and probability of packet loss. At one end of the characteristic scale are “best effort” protocols that do not guarantee any kind of service quality, at the other end of the scale are protocols that do guarantee a service quality, and then there are some protocols in

between that guarantee some specific attributes. *Number of Channels* indicates the number of channels the protocol uses to coordinate connection sessions between sending and receiving nodes. At one end of the characteristic scale are single channel methods and at the other end are multiple channel methods. There are some protocols that can operate using single or multiple channels depending upon the mode.

Table 2. MAC layer classification; those in italic were not used during simulations

Sub-protocol		C. Contention	D. Quality of service	E. Number of channels
MAC IEEE802.11	[8]	Resolution	None	Multiple
MAC IEEE802.11e	[11]	Resolution	Yes	Multiple
MAC IEEE802.16	*[9]*	*Resolution*	*Yes*	*Multiple*
MAC IEEE802.15.4	*[10]*	*Resolution*	*None*	*Single/Multiple*
CSMA	[12]	Random Access	None	Single
MACA	[12]	Resolution	None	Single
TDMA	[12]	Contention Free	None	Multiple
ALOHA	[12]	Random Access	None	Multiple

3.5 Network Layer Routing Characteristics

The routing protocols available within the simulation environment allow their classification to be simplified to three independent characteristics, as detailed in Table 3. The orthogonality of the alphabet characteristics allow the solution space to be represented by a three-dimensional vector space as shown in Fig. 5c. *Route Computation* specifies how the routes between nodes within the network are calculated. In this case, one end of the characteristic scale is represented by the reactive method whereby the route from source to destination is computed only at the point when data are to be sent. At the other end of the scale is the proactive method whereby routes to all nodes are pre-computed and the information is usually stored within a table. In-between these two characteristic extremes are methods where routes are partially pre-computed and partially computed when data are to be sent. *Update Period* specifies the method by which route information is updated. At one end of the characteristic scale is the event driven update such as a node entering or leaving the network. The periodic update where updates are carried out at pre-defined times regardless of the state of the network is at the other end of the scale. *Source Routing* defines how the routing information is transmitted across the network. At one end of the characteristic scale is the source method whereby the complete route is sent along with the data from the source node. The other extreme is the hop-by-hop method where only enough route information is sent with the data to traverse to the next node.

Table 3. Routing protocol classification

Sub-protocol	F. Route Computation	G. Update Period	H. Source Routing
OLSR-INIA [13]	Proactive	Hybrid	Hybrid
FISHEYE [14]	Proactive	Periodic	Hybrid
DSR [1]	Reactive	Event	Source
AODV [1]	Reactive	Event	Hop-by-hop
ZRP [14]	Hybrid	Periodic	Source
STAR [15]	Proactive	Event	Source

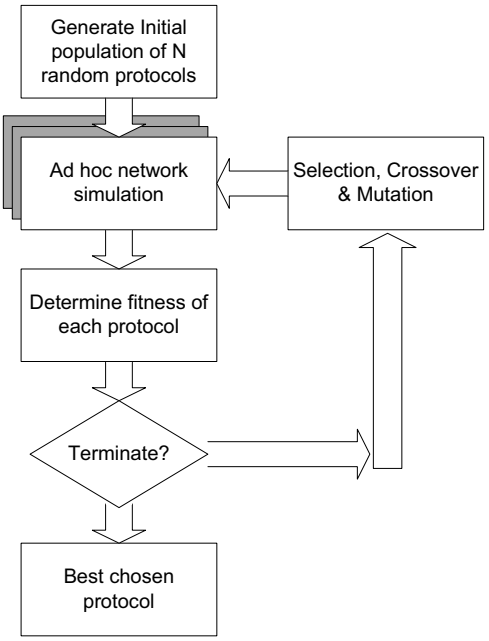
3.6 Interfacing Sub-Protocols

The decision regarding which sub-protocol to choose in each layer is carried out sequentially starting from the bottom physical layer. There are inevitably some sub-protocols that can only be interfaced to a subset of other sub-protocols in the next layer due to compatibility problems, leading to a reduced set of possible communication protocol stacks. After the choice of sub-protocol has taken place within the current layer, a simple masking method is used to reduce the available choice of sub-protocols at the next layer based upon the current layer's choice.

3.7 The Genetic Algorithm and Fitness Function

As shown in Fig. 3, this methodology uses a genetic algorithm [16] with an initial population of N random protocols which are simulated in turn, each returning performance measurements. These are then used by the fitness function to obtain a fitness score for each protocol. The selected fittest protocols then undergo crossover and mutation to create a new population of fitter protocols. This is repeated until an optimum solution is found. The aim of the genetic algorithm is to minimise a fitness function. The fitness function (F) is a sum of the chosen performance metrics which allow the network to be evaluated for a given protocol stack. The first performance metric (P1) is calculated within a defined period of time and given that the aim is to *minimise* the fitness function, the ratio of the two numbers is inverted from the normal calculation used for throughput. The subsequent three performance metrics (P2, P3 and P4) add a small penalisation factor for specifying a set of characteristic values a long way from the chosen protocol by taking the length of the shortest distance from the nearest protocol into account at each layer of the protocol stack. This is necessary due to the limited protocol choice meaning that some protocols took up a very large volume within

the solution space increasing the probability of being selected even when there were other equally fit protocol choices available.



$$F = P1 + P2 + P3 + P4. \quad (1)$$

$$P1 = \text{Number of packets sent} / \text{Number of packets received} \quad . \quad (2)$$

$$P2,P3,P4 = \text{Shortest distance in layer} / \text{Maximum distance in layer} \quad . \quad (3)$$

Fig. 3. Genetic algorithm flow

4 Network Scenario

A network scenario was generated to determine how well the methodology performed under changing network conditions by applying faults to the network and monitoring how the protocol adapted. Simulations were run five times for each scenario case generated. QualNet [17] was used for the operating environment and Matlab was used for the centralised controller. The protocol stack model used within QualNet closely resembles the OSI model and used a Constant Bit Rate (CBR) at the application layer, and User Datagram Protocol

(UDP) at the transport layer. The parameters for each of the sub-protocols were assigned their default QualNet values.

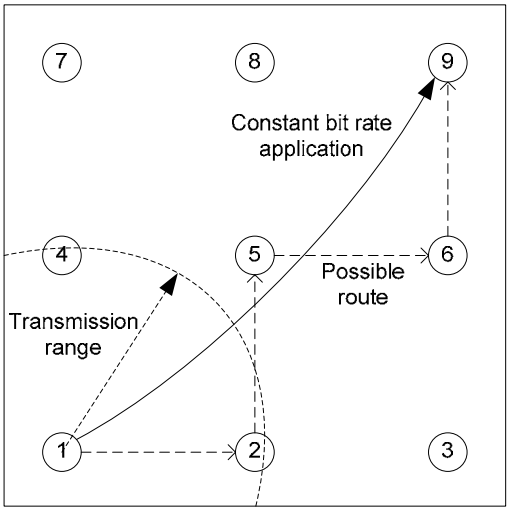


Fig. 4. Layout of the 3 x 3 mesh scenario

Within this scenario nine nodes were positioned in a 3 by 3 mesh arrangement as shown in Fig. 4. The distances between the nodes were set close to the maximum transmission range so that the probability of data packets transmitted diagonally, for example from node 1 to node 5, or even directly to node 9 was very low. This forced multiple possible routing paths when data was transmitted from node 1 to node 9 using a constant bit rate application. The simulation was run for 25 generations to find the optimum protocol. At the 25th generation faults were applied to the network and the simulation then ran for a further 25 generations to determine how the protocol adapted. The number of data packets received from each protocol stack combination was assessed independently to determine how well the algorithm performed. The simulation parameters for the scenario are given in Table 4. Three faults were applied to the network at nodes 2, 3 and 5. Intermittent faults were applied to nodes 2 and 3 whereby the faults prevented the node from operating for a short period of time at random intervals. A static fault was applied to node 5 which lasted for the first 5 seconds of the simulation.

Table 4. Simulation parameters

Parameter	Details
Sending node	1
Receiving node	9
CBR details	10Mbits/sec
Simulation time per protocol selected	15seconds
Population Size	25
Generation number when faults applied	25
Total generations	50
Fault 1	Node 2 intermittent
Fault 2	Node 3 intermittent
Fault 3	Node 5 static 0-5s
Maximum mobile speed of nodes	10m/s

Table 5. Scenario test cases

Case number	Mobility	Mutation rate
1	static	0.2
2	static	0.5
3	static	0.7
4	Random Waypoint	0.2
5	Random Waypoint	0.5
6	Random Waypoint	0.7

The scenario was run six times by varying two parameters. The first parameter, mobility, was set to either static where the nodes remained in a fixed position or set to random waypoint where the nodes could move about in a random fashion as an ad hoc network might behave in practice. The second parameter, mutation rate, was varied to investigate whether changing the diversity of the population was able to improve the ability of the algorithm to find the global optimum. Test cases are given in Table 5.

5 Results

Fig. 5 shows the output from a single run over 25 generations of the genetic algorithm under a mobile environment at the maximum mutation rate of 0.7 (first 25 generations of case 6 in Table 5), with no faults set. The crosses show the points generated by the genetic algorithm of chosen characteristic values. After 25 generations there is clustering around chosen protocols for each of the three optimised layers. For this particular case it correctly chose PHY 802.11a, CSMA, and FISHEYE as the optimum protocol selection.

Fig. 6 shows how the mean fitness score of the population changes over the generations. The mean fitness score diminishes quickly to a minimum at the 10th generation long before it approaches the 25th generation where the optimum protocol is established. After the 25th generation faults are applied and the mean fitness score rapidly increases as the current population is no longer optimal. At the 37th generation the mean fitness score diminishes again as the protocol adapts to the environment. For this particular case it correctly chose PHY 802.11a, MAC 802.11, and AODV as the optimum protocol selection.

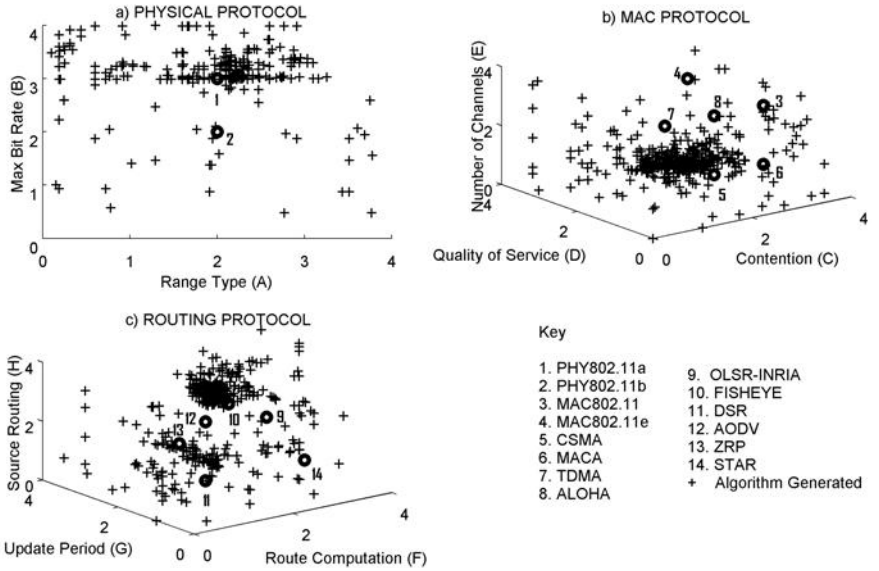


Fig. 5. Optimisation for the 3 by 3 mesh scenario for case 6 with no faults set

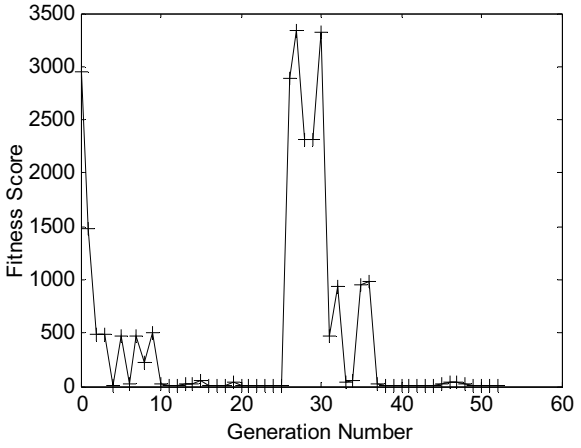


Fig. 6. Mean fitness score for a single run of case 6

Fig. 7 shows the number of times the correct protocol was generated over 5 runs for each of the 6 scenario cases. For the static node cases there appeared to be some improvement when a high mutation rate was used after faults were applied to the network. For the mobile case however the opposite was true and could be due to the fact that moving nodes is a harder problem to solve. Further testing would be needed before drawing more conclusions from these results. Out of the total 60 runs conducted for this scenario, 43 of them resulted in the identification of the correct optimum protocol suggesting a preliminary identification rate of 72%.

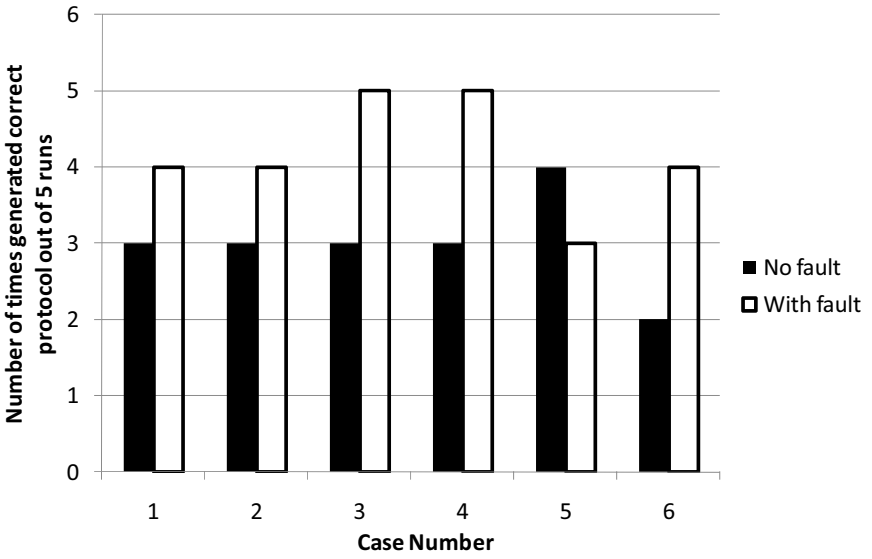


Fig. 7. Effect of varying the mutation rate on the ability of the genetic algorithm to find the optimum protocol for each of the 6 cases

6 Conclusion

The methodology proposed in this paper is a first step at dynamically evolving and adapting an ad hoc communication protocol under changing network conditions. It uses simple protocol mapping techniques and a genetic algorithm to select the optimum protocol for a given scenario using a simple fitness function to provide feedback regarding the network's current performance. Preliminary results show that the methodology is able to find global optima for a network scenario under varying conditions, and has a global optimum identification rate of 72%. The methodology is by no means complete and there are areas which can be developed

further. For example if the operating environment, which is currently simulated in QualNet, was directly replaced with a real-time environment then it would take a minimum of two and a half hours (plus computation and interfacing time) to establish an optimum protocol if all population trials were carried out in a sequential manner (25 populations x 15 seconds of run-time x 25 generations). This response time could be optimised down to a few minutes, making it more realistic, with higher data rates to capture throughput information for the fitness function in a shorter run-time, together with a fitness threshold to optimise and reduce the number of generations. Alternatively, or in addition to the above optimisation, the instantaneous state of the network could be captured at regular intervals and input into high speed offline parallel processors to predict the optimum protocol before sending a global protocol update and minimising disruption to the network. For realistic application within the distributed architecture of an ad hoc network however the methodology would need to be designed for real-time performance with distributed rather than centralised control. This would require each of the nodes acting as simple interacting elements evolving the optimum communication protocol through local interactions and decisions. Future work would include using FPGAs to provide this hardware architecture with parallel processing and run-time reconfiguration capability to allow dynamic protocol changes.

Acknowledgments

This work was supported by the Complexity Science Doctoral Training Centre at the University of Warwick under EPSRC funding. The authors would like to thank Professor Sadie Creese of the University of Warwick for helpful review comments.

References

1. E. Royer, and C.-K. Toh, “A review of current routing protocols for ad-hoc mobile wireless networks,” *IEEE Personal Communications Magazine,* vol. 6, no. 2, 1999.
2. P. Ocenasek, and M. Sveda, “An approach to automated design of security protocols,” in Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL’06), 2006.
3. H. Xue, H. Zhang, and S. Qing, “A schema of automated design security protocols,” in International Conference on Computational Intelligence and Security Workshops, 2007.
4. A. Perrig, and D. Song, “On a first step to the automatic generation of security protocols.”
5. S. Virtanen, J. Isoaho, T. Westerlund, and J. Lilius, “A programmable general protocol processor - a proposal for an expandable architecture,” in *URSI/IEEE XXIV Convention on Radio Science*, 1999.

6. J. Oberg, A. Kumar, and A. Hemani, “Scheduling of outputs in grammar-based hardware synthesis of data communication protocols,” *IEEE*.
7. I. Pavlosoglou, M. S. Leeson, and R. J. Green, “Applying emergence to the design of routing protocols for the security of wireless ad hoc networks,” in Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks, 2005.
8. R. Jurdak, C. V. Lopes, and P. B. Baldi, “A survey, classification and comparitive analysis of medium access control protocols for ad hoc networks,” *IEEE Communications Surveys,* vol. 6, no. 1, 2004.
9. F. Wang, A. Ghosh, C. Sankaren *et al.*, “Mobile wimax systems: Performance and evolution,” *IEEE Communications Magazine,* vol. 46, no. 10, 2008.
10. E. D. Pinedo-Frausto, and J. A. Garcia-Macias, “An experimental analysis of zigbee networks,” in IEEE Conference on Local Computer Networks, 2008.
11. E. Ferro, and F. Potorti, “Bluetooth and wi-fi wireless protocols: A survey and a comparison,” *IEEE Wireless Communications,* vol. February, 2005.
12. A. C. V. Gummalla, and J. O. Limb, "Wireless medium access control protocols," *IEEE Communications Surveys & Tutorials,* Second Quarter, 2000.
13. P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, and L. Viennot, “Optimized link state routing for ad hoc networks,” in Technology for the 21st Century Multi Topic Conference, 2001.
14. X. Zou, B. Ramamurthy, and S. Magliveras, “Routing techniques in wireless ad hoc networks - classification and comparison,” in Proceedings of the Sixth World Multiconference on Systemics, Cybernetics, and Informatics, 2002.
15. J. J. Garcia-Luna-Aceves, and M. Spohn, “Source-tree routing in wireless networks,” in Seventh International conference on Network Protocols, 1999.
16. D. Whitley, “A genetic algorithm tutorial,” *Statistics and Computing,* vol. 4, 1994.
17. "Qualnet." http://www.scalable-networks.com.

The Next Generation of Legal Expert Systems - New Dawn or False Dawn?

C. Stevens[1], **V. Barot**[2], **and J. Carter**[3]

Abstract Attempts to apply conventional rule-based expert systems to legal problem-solving raise seemingly insurmountable obstacles. The authors analyse the key challenges of developing a legal expert system by reference to a case study of issues arising in their prototype system, the JAES project. This paper explores the advantages of exploiting three alternative approaches- namely: case-based reasoning, blackboard architecture, and service-oriented architecture for the next generation of legal expert systems. The authors advocate the use of hybrid architecture to address the complexity and dynamic nature of the legal domain. The paper evaluates the extent to which these enhancements can meet the special complexities of the legal domain.

1 Introduction

Are reports of the demise of legal expert systems greatly exaggerated? Since the 1990's, commentators have noted that interest has diminished in legal expert systems in favour of other applications of information technology to legal practice that are less challenging and offer more immediate returns (such as automated document assembly) [1]. However, the utilisation of case based reasoning and the exploitation of blackboard and service oriented architecture offer the prospect of developing a new generation of better and more sophisticated legal expert systems. This paper will explore whether the next generation of systems will be capable of surmounting the unique challenges presented by legal problem-solving. If these techniques can be exploited successfully a resurgence of interest in the

1 Leicester Institute of Legal Practice, De Montfort University Law School, UK
cstevens@dmu.ac.uk

2 Wolfson Institute, University of Loughborough, UK
vbarot@lboro.ac.uk

3 Centre for Computational Intelligence, De Montfort University, UK
jennyc@dmu.ac.uk

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_33,

development of legal expert systems can be expected. Anyone with an interest in artificial intelligence or legal practice needs to consider these issues and join the debate.

The motives for applying expert system techniques to legal practice are self-evident. A list of the possible benefits [2] include: faster delivery of legal advice; liberation of fee earning time otherwise spent in the labour-intensive and repetitive tasks of taking instructions; carrying out legal research and giving advice; increased productivity; reduced dependence on transitory human expertise; potential savings in staff overheads; increased scope for delegation of tasks to a lower grade of fee earner; reduction in human error leading to improved claims record and lower insurance; cost savings arising from the above; increased profitability for the provider of legal services; and from the client's perspective, lower fees.

Conversely, the disruption to the traditional model of providing bespoke legal services; and the reduction in the time cost element of billing will be seen by many legal service providers as potential disincentives. However, it has been argued that the future viability of time costing as a basis for billing is likely to come under pressure as a result of increasing competitive pressure in the market for legal service [1]. It is submitted this market pressure will tip the balance in favour of efficiency savings [3]. Some commentators have argued that the 2 disciplines are fundamentally incompatible [4]. Whilst there have been some successful applications in the legal domain [6], these have been in relatively narrowly-defined and selected areas of the law. It has also been observed that the extent of commercial exploitation in the English legal profession is disappointing in comparison with other professional sectors and industry [6, 7, and 3].

This paper analyses these issues and places them in a practical context. This contextual approach will refer to issues arising from a particular prototype application of a legal knowledge-based system designed and developed by the authors. The prototype was given the name the Judicial Advisory Expert System project abbreviated to JAES [8]. The analysis will follow the following structure:

Section 2 will give a concise summary of the JAES project. Section 3 of the paper will analyse the particular challenges raised by legal expert systems with reference to the JAES project. Section 4 will examine the exploitation of case-based reasoning and the blackboard and service-oriented architectures. It will then explore the possibility of hybrid integration of these architectures and methodologies. Section 5 will evaluate the extent to which the next generation of systems will be able to meet the challenges.

2 The Jaes Project

The JAES project in essence aimed to embody the rules on passing of property and risk that are contained in sections 16 to 20 Sale of Goods Act (SGA) in a

prototype expert system. Disputes between buyers and sellers on the question of which party bears the risk of accidental damage to goods are inevitable. Insurance companies need to clearly identify the risk-borne party when settling claims. Traditionally, in such a situation, sections 16- 20 of the Sale of Goods Act (SGA) are applied by lawyers who advise the parties or their respective insurers. The traditional approach is both time consuming and expensive. JAES was developed by the authors in order to speed up the process and make it more cost effective especially for low value and routine disputes. The inference engine applies the logic contained in the knowledge base to the information input by the user and outputs advice on whether the seller or the buyer bears the risk of loss or damage to goods in a particular contract for the sale of goods. The rules comprised in sections 16 to 20 SGA were deliberately selected as the subject matter for the knowledge base, as the authors perceived them to be an attractive area for a legal expert system for the following reasons: the logical basis underlying sections 16 to 20 SGA is relatively clear; it is an area of the law of practical commercial importance; this area of law has been stable with relatively infrequent statutory amendments. A modular approach to the design of the system was adopted. Owing to the number of rules required, the system's performance needed to be enhanced in a later version by utilising a Q-learning algorithm. This modular design with Q-learning algorithmic implementation is described in [9]. The key output of the system is to use sections 16 to 20 SGA in the knowledge base to advise a user whether the seller or the buyer is responsible for bearing the loss of damage to goods.

3 What are the particular challenges of the legal domain?

3.1 Complexity

The JAES project selected a relatively discrete, narrow and specific area of law-namely the rules comprised in sections 16 to 20 of the Sale of Goods Act and applicable case law precedents. However, the reality of legal problems is that they are seldom confined to discrete and narrow areas of law. In practice they usually raise multiple issues of law which are interconnected and therefore cannot be isolated [10].

3.2 Uncertainty

Rule-based expert systems rely on clear factual premises for the rules in the inference engine to work on. However, legal reasoning in practice is by its very nature adversarial.

Law in any jurisdiction is not static. Therefore any legal expert system has to be flexible and constantly updated to take account of changes emanating from legislation, statutory instruments, case precedents, or European Union law. A possible approach to the updating of a system to take account of case law precedents is to use case-based reasoning [13]. Case based reasoning inferences by analogy. In essence it derives principles from specific cases and applies them to subsequent cases. However, in legal applications it will be challenging to isolate the important principle established by the precedent (referred to by lawyers as the ratio decidendi) and distinguish this from irrelevant factors that do not form part of the precedent. The integration of case-based reasoning with rule-based reasoning in legal knowledge-based systems is one way of addressing this problem [14].

3.3 Financial Disincentives

The investment of time required to develop a knowledge- based system even in a narrow area of law is considerable. Firstly, there is the onerous task of deriving the domain knowledge and structuring this in a manner which is comprehensible to the knowledge engineer. Once derived the knowledge and expertise has to be translated into computer code. Testing improving and updating the system also needs to be factored into the time required. Owing to law's complexity and open-texture a legal expert system requires more time than many other domains. From the point of view of a legal practitioner time represents lost fee earning time measurable in financial terms [15].

From the perspective of the end user it is not likely to be cost effective to invest in the purchase of an expert system to resolve a particular legal problem that may only arise infrequently. However, a web services application which is only accessed and paid for on a pay for use basis makes more economic sense.

4 Alternatives for the next generation of legal expert systems

In this section, case based reasoning (CBR) as an appropriate alternative or addition to a rule based system is proposed. Additionally, Blackboard and Service-oriented architectures are considered as having particular relevance for the overall

structure of the next generation of legal expert systems. Both architectures are described prior to illustrating their hybrid integration for the legal expert system paradigm.

4.1 Case Based Reasoning (CBR)

The design of most intelligent systems is usually inspired by the desire to emulate human capabilities in some way. One example of this is the way that humans are able to recall previous similar experiences when attempting to solve problems [19]. CBR is an approach to intelligent systems development that aims to emulate this capability. A CBR system stores problems, retrieves closest matches when a new problem needs to be solved, adapts the previous approach to suit the new problem exactly and finally stores the new case. The stored knowledge is known as the case base and one way of storing these is as objects with links between classes and instances that enable effective retrieval to take place. Such an approach might be more appropriate in the legal domain. One significant reason for this is that the knowledge base is built up over time, thus getting over the "knowledge acquisition bottleneck" [15] to some extent; secondly it can evolve in much the same way that the law does, so the content need not be artificially static.

Rissland, Ashley and Branting [27] consider the historical development of CBR and law identifying a number of systems that have exhibited varying degrees of success in different areas of the law and using different mechanisms for reasoning. A notable example is the HYPO system [28] that applies case-based reasoning in the legal domain using a 6 stage process. HYPO's use of CBR has the considerable advantage for the legal domain of outputting alternative arguments as opposed to definitive answers. This provides a more realistic outcome which is more reflective of the open-textured nature of legal problem solving. HYPO's system has been utilised in a more recent system known as CATO [29,30,31] to teach law students how to reason with case law precedents in the law relating to trade secrets and confidential commercial information. Other systems that are developments of HYPO are described in [27].

ASHD-II [32] was developed as a hybrid legal system in the area of divorce law. It consisted of a rule base and a case base. The reason for the development of this hybrid was to take advantage of both of these methods since the nature of law means that it can be necessary to use precedents (easily represented by CBR) and other legal sources such as statutes, codes (more easily represented in a rule based format). The system illustrated that there was success in terms of creating the hybrid system but that even this did not did not fully capture the behaviour of a legal practitioner and they concluded that the system was more useful as an aid to the less experienced practitioner. Law provides an excellent testing ground for new approaches to CBR and by its nature CBR lends itself very well as a potential mechanism for automating some areas of legal decision making.

This paper proposes that CBR be used to develop one or more of the knowledge sources required for the blackboard architecture; the other knowledge sources would be rule based. The most appropriate mechanism would then be applied for each problem that the system is required to handle. The blackboard architecture is explained in the next section.

4.2 Blackboard architecture

This is a task-independent architectural design which mimics the natural process where heterogeneous team members solve a particular problem via a communication medium called a blackboard. Blackboard architecture is informal, owing to the fact that it can be applied to a variety of problems and each problem involves a slight reinterpretation of the architecture itself [16]. It has been viewed as an ideal architecture for open-textured, complex or non-deterministic problems [18]. It promises therefore more sophisticated techniques which are more appropriate for legal problems. The concept of this architecture is shown in figure 1 and it is based on the metaphor of a meeting room, where a number of different experts surround a discussion board and use their expertise cooperatively to brainstorm a particular complex problem posted onto the board. The discussion board equates to the "blackboard" component of this architecture. The "knowledge source" component equates to each specialist who contributes their particular expertise in solving the problem. In order to control the flow of the problem solving process and schedule the contributions of each of the knowledge source onto the discussion board, the "controller" component of the blackboard architecture is needed which is equivalent to a chair in the human metaphor.

The knowledge sources (KS) share a common global dynamic database (i.e. the blackboard). The access to this shared resource is managed by a control shell (i.e. controller). The knowledge sources can be either internal or external (i.e. remote). The blackboard can be a single publicly accessible region or subdivided into regions or panels. The controller can be implemented as a separate entity (centralised) or can be partly implemented in the blackboard and partly in the knowledge sources (distributed). Communication between the knowledge sources can only take place via the blackboard.

The problem solving scenario begins when an initial problem is posted onto the blackboard by a knowledge source. The knowledge source can be as small as a system function or as large as a complete expert system. The posted problem is globally accessible through the dedicated memory area (i.e. the blackboard) whose controller is responsible for triggering the specialist knowledge source to contribute its solution towards solving the posted problem. Once the problem is cooperatively solved by all the knowledge sources, the next problem can be generated for continuous applications. This methodology corresponds to the way human beings solve problems in a distributed team. This architecture is a highly

modular way of building problem solving systems [18]. Modularising the components allow interactions between them to be regularised [20]. Furthermore, it allows clear and rigid interfaces to be defined through which the components can be accessed. Each of the components of this methodology offer modularity as well as other significant system-level benefits including performance, re-usability, security, maintainability and reliability of the overall system.

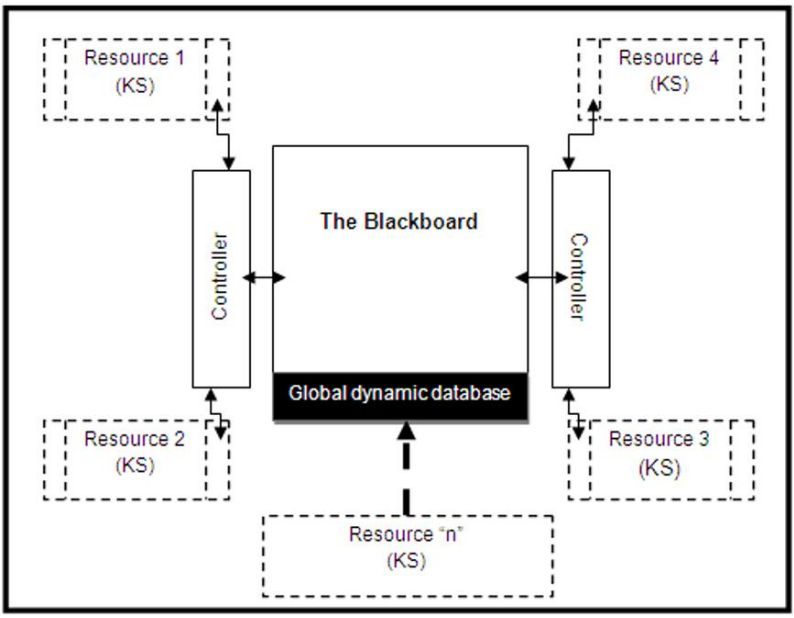


Figure 1: Blackboard Systems Architecture

The problem solving scenario begins when an initial problem is posted onto the blackboard by a knowledge source. The knowledge source can be as small as a system function or as large as a complete expert system. The posted problem is globally accessible through the dedicated memory area (i.e. the blackboard) whose controller is responsible for triggering the specialist knowledge source to contribute its solution towards solving the posted problem. Once the problem is cooperatively solved by all the knowledge sources, the next problem can be generated for continuous applications. This methodology corresponds to the way human beings solve problems in a distributed team. This architecture is a highly modular way of building problem solving systems [18]. Modularising the components allow interactions between them to be regularised [20]. Furthermore, it allows clear and rigid interfaces to be defined through which the components can be accessed. Each of the components of this methodology offer modularity as

well as other significant system-level benefits including performance, re-usability, security, maintainability and reliability of the overall system.

4.3 Service-oriented architecture (SOA)

This architecture provides a model to develop systems that assemble and distribute services remotely. The key advantage of service-oriented architecture (SOA) is that it enables a service provided by a third-party to be directly accessible to other systems regardless of their mechanisms (e.g. operating systems or application software) or their geographic locations. The term "service-oriented" emphasises the fact that the main priority of this architecture is centred on the service provided. Constraints and obstacles that hinder accessibility of the service in other architectures are made subservient to the service and are prevented from hindering access to the service. Businesses therefore that wish their services to be accessible and to reach a wider market understandably find this architecture commercially attractive. Consequently, SOA has been widely exploited in a number of service sectors such as financial services, and travel and tourism services, including holiday bookings. SOA achieves this accessibility by using web services (WS) representation. Figure 2 illustrates the basic architectural representation of web services incorporating the operations, entities and the components necessary for its functionality. These are 'Service Provider', 'Service Registry' and 'Service Requester [20, 21].

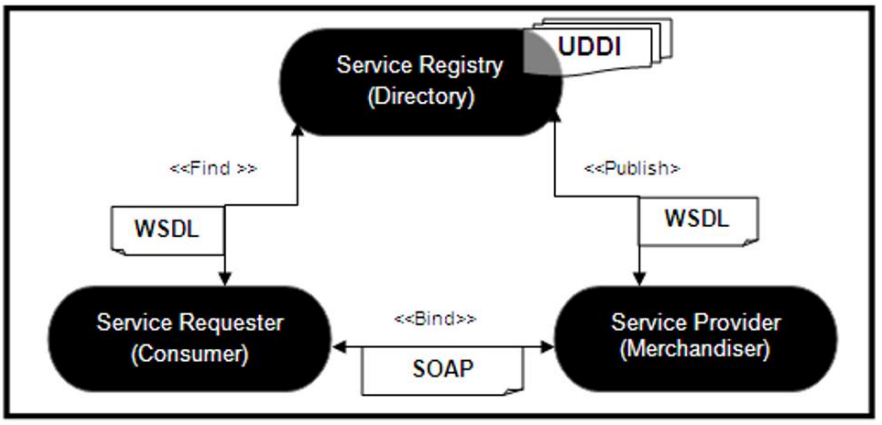


Figure 2: Web services Representation Model

The 'Service Provider' designs a particular service by defining its interface clearly and implementing its service functionality. The provider describes and

publishes its details using WSDL into a 'Service Registry' [17]. WSDL provides an XML description of what messages can be exchanged and a point of contact for a successful interaction. XML messages are human and machine readable, which facilitates the debugging process and provides uniformity. The 'Service Registry' is a generally accessible registry (directory) which uses a publication standard called UDDI. The UDDI provides a directory service where all the information about a particular web service such as the service provider, type of the service, location of the service description and other business related information is kept. The UDDI contains a list of web services offered by various providers and can be searched by the 'Service Requester'. The registry can be public, private or restricted, which enables one to keep track of the currently available web services [21].

The 'Service Requester' is a consumer of the web-service who locates a specific web service by querying it from the UDDI registry. Once the description and the specification of the web service are discovered, the requester binds its application to the specific service required and communicates with it using a protocol called SOAP [22]. SOAP is an XML-based communication protocol which provides the envelope for sending web service messages using HTTP (Hypertext Transfer Protocol). The requester and the provider can be implemented in an application locally or remotely. In order to ensure compatibility and accessibility by other systems, a web service interface needs to be defined by the provider and published by the registry. This interface possesses the necessary definitions and the methods to either access any underlying data or carry out specific tasks [23].

The authors foresee a number of advantages in utilising service oriented architecture for legal expert systems in preference to conventional approaches. These are described below.

Since the web services interactions are built on existing standards like XML, HTTP, WSDL, UDDI and SOAP, it can resolve some of the limitations related to web-based expert systems. Firstly, the standardisation of message transfer can be solved using XML. Secondly, compatibility among communication protocols of various systems over the Internet can be tackled using HTTP. Furthermore, bypassing existing security infrastructures such as a firewall can be confronted using the existing HTTP business infrastructure, etc.

Enterprises relying on systems using earlier technology (referred to as legacy mechanisms) need not discard the functionality offered by these systems. Web services can easily be wrapped around legacy systems, thus requiring little or no change to the existing mechanisms. This offers the benefit of distributing knowledge to a wider range of remote audience using the existing mechanisms by plugging them in to the web services framework.

Web services enable loose-coupling of software components provided the interface has been defined and specified. This will enable the knowledge base to be amended at one end without affecting the functionality of the service delivered at the other end. As expert systems invoke web services at run-time, any upgrade

to the underlying knowledge exposed as a web service, does not hinder the execution of the system as long as the interface accurately reflects its description.

Updating an existing legal expert system to reflect changes in the law is facilitated by the modular framework offered within the web services paradigm. This provides opportunities for reducing the time and expense incurred in manually updating the existing legal knowledge with the newer knowledge within an expert system.

Web services-based implementation offers income generation revenue for a legal enterprise offering the necessary knowledge as a service. An end-user can be charged according to the type of legal advisory service requested. This is more cost-effective than conventional approaches, where users purchase and install license components of a system which they may never use or only access infrequently. This also provides an opportunity for legal advisory firms and consultancies of any size to invoke web services-based legal knowledge. Commoditisation of systems is enhanced and supported by web services standards. The increased opportunities could make the next-generation of legal expert systems more commercially attractive for lawyers to develop and for users to access.

While conventional expert systems are limited in their narrow domains, web services-based systems offer the potential to access external systems remotely to deal with other specialist areas.

4.4 Hybrid integration

Limitations have been identified by Grove [24] and Duan [25] with existing web-based expert systems such as interface complexity, limited infrastructure, communication loads, inference complexities, to mention only a few. The authors foresee the next generation of legal expert systems supporting day-to-day legal advisory business requirements using a systematic and extensible framework for application-to-application interaction. This application-to-application interaction grants efficient mechanisms for large amounts of data to be transferred from one global point to another and specific tasks to be undertaken, automatically without the need for data to be processed by a web browser [26]. Globalisation demands a solution that can provide access to detailed legal expertise covering different specialist areas for effective decision making. The complexities, resource allocations (time and money) and the available technology prevent expert systems from processing vast amounts of knowledge in a stand-alone legal expert system implementation. In addition, legal expert systems need constant modification as and when the law is modified or repealed. The solution to these key issues is a hybrid integration of blackboard architecture with the flexible framework offered by web services-based SOA.

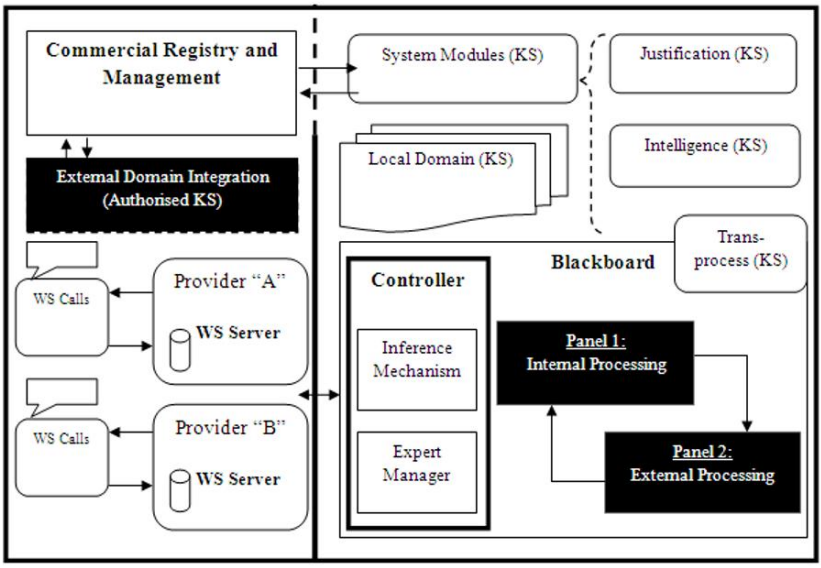


Figure 3: Hybrid Architectural Design

This hybrid architectural design is shown in figure 3. The common data structure, (i.e. the blackboard) stores initial data, any intermediate decisions and the final solution. In order to minimise complexity in processing the knowledge bases and to improve the performance of the overall system, the blackboard is divided into two distinct areas called panel 1 and panel 2. Panel 1 deals with internal processing associated with any local legal domain knowledge of the expert system; whereas, panel 2 deals with all the external processing associated with any web service invocations. Any information transfer between panel 1 and panel 2 is carried out using a dedicated knowledge source (KS) called trans-process. The controller is divided into two logical sections namely an inference mechanism and an expert manager. The inference mechanism implements a reasoning technique of the expert system whereas the expert manager controls and manages all the access and contributions made by every knowledge source at runtime.

The knowledge sources can be domain specific or system specific. Domain specific knowledge sources are further divided into local legal domain knowledge sources, pre-existing in the expert system, and external authorised legal domain knowledge sources, dynamically acquired using web services invocation. System specific knowledge sources are modules which contribute to the overall workability of the system and provide add-on services like justification, online / offline support and assistance, system intelligence, etc. The architecture stresses the importance of a commercial registry and management legislation system to be in place to link the internal expert system knowledge processing mechanism with the SOA based legal knowledge bases, in order to support necessary web service

registration, authorisation, validation and pricing control procedures. The authors advocate a hybrid architectural design style shown in the figure 3 to achieve the sophistication and flexibility needed in a legal expert system.

5 Will the next generation of systems overcome the challenges?

Blackboard architecture and the scope it gives for using different reasoning techniques including both rule based and case based and multiple expert systems in a collaborative manner is a better way of attempting to use artificial intelligence to deal with the multi-faceted, interconnected, and open-textured nature of legal problems than traditional methods that rely exclusively on single systems confined to a highly selective discrete area of law. Blackboard architecture also offers a better solution to legal updating through its inherent modularity and the flexibility deriving from its lack of dependence on one system.

Service oriented architecture for its part offers the next generation of systems the full unrestricted potential of the web both to access external sources of knowledge and expertise and to make the expertise available to a global market. The advanced techniques of service oriented architecture for overcoming the technical obstacles of compatibility open up new dimensions for exponential development of a new generation of legal expert systems. The prospect of accessing multi-jurisdictional legal expertise in a globalised economy will be demand led. Updating web based systems is considerably less cumbersome than traditional methods. Legal service providers that do not have the time or resources to update systems have the option available to outsource this function to external sources of expertise.

Hybrid integration of blackboard architecture with SOA combines the advantages of both designs to considerably improve functionality in the next generation of legal expert systems. Internal multiple expert systems using blackboard architecture can also access as and when required unlimited external sources paid for on a software use basis. The prospect of delivering legal expert advice digitally by remote means to an unlimited number of users free of technical barriers makes the whole process of expert system development more economically attractive and the financial disincentives less of a barrier. Enhanced cost effectiveness in paying for expertise only when it is required will lead to increasing demand for cheaper and faster commoditised solutions in the next generation of expert systems. In a highly competitive and challenging legal market place this client driven demand may prove irresistible.

To address the question implicitly raised in the title the advances in web technology and hybrid applications of blackboard architecture and SOA are likely to result in the dawn of a new era of development and growth in the next generation of enhanced legal expert systems. The next generation will see significant improvements on current systems by utilising the techniques analysed

in this article. Untapped potential for their application in selected areas of legal practice will be realised by competitive market pressures and technical progress. However, the next generation of legal expert systems will not be able to surmount all of the problems raised by the challenging, open-textured, and dynamic domain of law. In particular the problem of uncertainty of facts and the uncertain application of the law to particular problems cannot be totally eliminated. The appropriate role for the next generation of legal expert systems will be to supplement and augment traditional legal advice. Although the total replacement of the lawyer with a digital machine may be the desired objective of many, the unique challenges of the legal domain will continue to prevent the practical realisation of this aspiration even in the next generation of legal expert systems.

References

1. Susskind, R. 'The End of Lawyers? Rethinking the Nature of Legal Services' 2008, OUP
2. Jenkins, J. 2008, 'What can information technology do for law?' Harvard Journal of Law & Technology Volume 21, Number 2 Spring 2008
3. Bibel, L.W. 'AI and the conquest of complexity in law' Artificial Intelligence and Law 2004 Volume 12, 167
4. Leith, P. 'fundamental Errors in Legal Logic Programming' The Computer Journal, Vol. 29, 545-552
5. Susskind. 2000 *Transforming the Law* Oxford University Press
6. Oskamp and Lauritsen 'AI in law and practice? So far, not much' Artificial intelligence and law. 2002 Volume 10 pages 227-236
7. Oskamp,Tragter and Groendijk, 'AI and Law: what about the future?' Artificial Intelligence and Law Volume 3 Number 3 September 1995 pages 209-215
8. Barot, V., Carter, J. 2008, Design and Development of a Judicial Advisory Expert System (JAES) to Resolve Legal SGA Ownership Dispute Cases, The 2008 UK Workshop on Computational Intelligence, UKCI 2008
9. Barot, V., Carter, J. 2008, Implementation of the Q-Learning Algorithm for Optimising a Judicial Advisory Expert System (JAES), UKCI 2008.
10. Yannopoulos, G.N., 1996, Modelling the legal decision process for information technology applications in law, PhD thesis, 1996
11. Von der Lieth Gardner, A., 1987, *An Artificial Intelligence Approach to Legal Reasoning* 1987 Massachusetts Institute of Technology
12. Prakken, H., and Sartor, G., 1998, 'Modelling Reasoning with Precedents in a formal Dialogue Game, Artificial Intelligence and Law, Vol. 6, 231-287
13. Wyner, A., 2008, 'An Ontology in Owl for Legal Case- Based reasoning' Artificial Intelligence & Law, vol. 16, 361-387.
14. Vossos, G., 1991, 'An Example of Integrating Legal Case Based Reasoning with Object-Oriented Rule-Based Systems: IKBALS II available online.
15. Buchanan, B. G., and Feigenbaum, E. A. 1982. Forward. In Davis, R., and Lenat, D. B., eds., Knowledge-Based Systems in Artificial Intelligence. McGraw-Hill.
16. Craig, I.D., Blackboard systems. Artificial Intelligence Review, 1988. 2(2), 103-118.
17. Corkill, D.D. Collaborating software: Blackboard and multi-agent systems & the future. 2003.
18. Sommerville, I, Software Engineering. 8th ed. 2007, Reading, Massachusetts: Addison-Wesley Publishing Company.

19. Hopgood, A. 2001, Intelligent Systems for engineers and Scientists, CRC Press.
20. Cerami, E., Laurent, S.S., 2002, Web services essentials. O'Reilly & Associates
21. Potts, S. and M. Kopack, 2003, teach yourself web services in 24 hours, SAMS
22. Newcomer, E., 2002, Understanding Web Services: XML, Wsdl, Soap, and UDDI., Addison-Wesley.
23. Coyle, F.P., 2002, XML, Web services, and the data revolution. Addison-Wesley Longman, USA.
24. Grove, R.F., Design and development of knowledge-based systems on the web. In: Proceedings of ISCA 2000: Ninth International Conference on Intelligence Systems: Artificial Intelligence Applications for the New Millennium, International Society of Computer Applications (ISCA), 147-150.
25. Duan, Y., J.S. Edwards, and M.X. Xu, *Web-based expert systems: benefits and challenges.* Information & Management, 2005. 42(6), 799-811.
26. Newcomer, E., Understanding Web Services: XML, Wsdl, Soap, and UDDI. 2002: Addison-Wesley.
27. Rissland E. Ashley, K, Branting, K. 2006, Case Based Reasoning & Law, The Knowledge Engineering Review, vol. 20:3, 293-298l
28. Rissland E., Valcarce, E., Ashley, K, 1984, Explaining and arguing with examples-Proceedings of Fourth National on Artificial Intelligence. AAAI Press, 288-294.
29. Aleven, V., 1997, Teaching case-based argumentation through a model and examples, PhD thesis, University of Pittsburgh
30. Aleven, V., 2003, Using background knowledge in case-based legal reasoning: A computational model and an intelligent learning environment, Artificial Intelligence,Volume 150, Issues 1-2, November 2003, Pages 183-237
31. Aleven, V., & Ashley, K. D. (1997). Teaching Case-Based Argumentation Through a Model and Examples: Empirical Evaluation of an Intelligent Learning Environment. In B. du Boulay & R. Mizoguchi (Eds.), Artificial Intelligence in Education, Proceedings of AI-ED 97 World Conference (pp. 87-94). Amsterdam, The Netherlands: IOS Press
32. Pal, K., & Campbell, J., 1997, An Application of Rule-Based and Case-Based Reasoning within a Single Legal Knowledge-Based System, The Data Base for Advances in Information Systems, vol. 28, 4.

Incorporating Semantics into Data Driven Workflows for Content Based Analysis

M. Argüello and M.J. Fernandez-Prieto[1]

Abstract Finding meaningful associations between text elements and knowledge structures within clinical narratives in a highly verbal domain, such as psychiatry, is a challenging goal. The research presented here uses a small corpus of case histories and brings into play pre-existing knowledge, and therefore, complements other approaches that use large corpus (millions of words) and no pre-existing knowledge. The paper describes a variety of experiments for content-based analysis: *Linguistic Analysis* using NLP-oriented approaches, *Sentiment Analysis*, and *Semantically Meaningful Analysis*. Although it is not standard practice, the paper advocates providing automatic support to annotate the functionality as well as the data for each experiment by performing semantic annotation that uses OWL and OWL-S. Lessons learnt can be transmitted to legacy clinical databases facing the conversion of clinical narratives according to prominent Electronic Health Records standards.

1 Introduction

Recognition of meaningful data in clinical narratives requires a level of expertise that only expert clinicians possess. The differences in knowledge representation between expert and novice are more evident in highly verbal domains. Psychiatry stands out as a highly verbal domain, where the effects of expertise when comprehending psychiatric narrative have been investigated. Studies performed, such as [1], revealed that non-experts were less able to distinguish relevant from irrelevant information, and the inference made by them were less accurate. This corroborates that the recognition of meaningful associations between text elements and knowledge structures is far from trivial in psychiatry.

The research study presented here recognises that a complete semantic understanding from clinical narratives in highly verbal domains, such as psychiatry, emerging from a computer-based content analysis is a challenging

1 University of Salford, M5 4WT, UK
m.arguello@computer.org

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_34,

goal. However, it is worthwhile investigating what can be achieved towards this goal by means of Natural Language Processing (NLP) or pre-defined lexicons.

There have been recent studies [2][3] that have considered the use of information technologies to support mental health. The research study presented is aligned with [2] because it also looks at the problem of finding meaningful associations between text elements and knowledge structures within psychiatric narrative. While in [2] a large corpus of 30 million words and no pre-existing knowledge are used; the current study uses a small corpus of 5716 words and brings into play pre-existing knowledge. Therefore, [2] and the current study look at the problem from different but potentially complementary angles.

Nowadays, the growing number of available lexical resources offers the possibility of using these lexical resources to perform experiments that aim to enhance content-based analysis. The current research follows a service-oriented approach, where the experiments performed encourage service-reuse and *service composition* (the process of combining different services to provide a *value-added service*). Unlike business workflows that are often event-flow driven, scientific workflows are generally data-flow driven (i.e. execution is based on the flow of data as opposed to triggered events) [4]. To keep track of the services and data for each experiment, the research presented applies semantic annotation. This is not standard practice. Although, the creation of a repository that stores the functionality and also the data, which is needed for each service or exchange among services, facilitate experiments replication and encourage resource (either service or dataset) reuse; see how myExperiment [5] promotes sharing workflows.

There is a practical value from this research that can be transmitted to legacy clinical databases that need to be transformed according to prominent Electronic Health Records (EHRs) standards such as EHRcom [6], openEHR [7], or HL7 CDA [8], and where these legacy systems have to face converting clinical narratives into a formal representation that is easy for computers to manipulate.

The paper is structured as follows. Section 2, 3, and 4 look respectively at content-based analysis from different angles: *Linguistic Analysis* using NLP-oriented approaches, *Sentiment Analysis*, and *Semantically Meaningful Analysis*. Section 5 presents the performance of content-based analysis for the more promising approaches. Section 6 provides insights into the semantic annotation performed. Concluding remarks are in section 7.

2 Linguistic Analysis Using NLP Oriented Approaches

Terms are usually referred to as the linguistic surface manifestation of concepts. Since biomedical literature is expanding so dynamically, the demand from the user community is directed towards practical and useful systems that are able to identify and link relevant “entities”. In recent years, a number of frameworks that

support ontology learning processes have been reported, see [9] to know more about the state of the art and open issues.

OntoLancs [10] and Text2Onto [11] are recent ontology learning frameworks particularly promising as both of them have been designed with a workflow editor that allows the combination of various algorithms from NLP and Machine Learning (ML). Both of them allow representing the learned ontological structures in a concrete knowledge representation language, such as OWL [12]. However, none of them uses pre-existing domain knowledge or provide automatic support to annotate the workflows that entail different combinations of methods, although OntoLancs [10] has made efforts towards this aim [13].

The current research fosters experimentation and aims to formally annotate the results obtained with different combinations of resources (either services or datasets) as well as the workflows themselves. Hence, this study pursues the document-centred view for NLP-oriented approaches proposed by [14] that outlines the use of OWL-S [15].

Figure 1 shows three pipelines that illustrate dynamic combinations of NLP-oriented approaches that have been used to conduct three experiments. The implementation of each modular component (box) as a service allows the creation of workflows, in other words, to perform service composition.

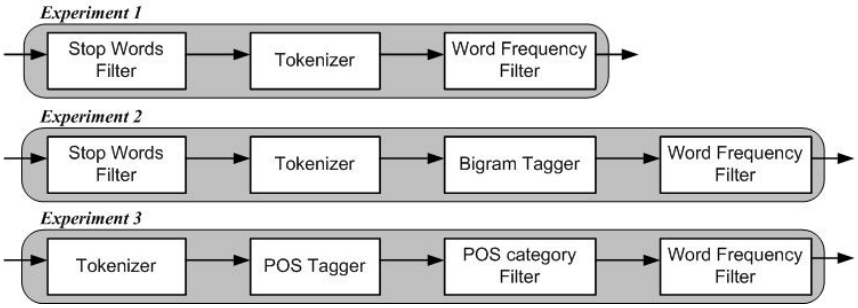


Figure 1 Three pipelines that exemplify combinations of NLP-oriented approaches

The starting point of the research study is a small size corpus (5716 words) composed of 25 case histories; all of them are available online. This number does not differ from the number of patients reported in psychiatric studies, such as [16][17]. The corpus is divided into two: 20 case histories for *training* and 5 case histories for *testing*. It is expected that experiments with the training corpus using NLP-oriented approaches may bring few useful terms, i.e. low *precision*. Nevertheless, these experiments can provide valuable insights into how to perform semantic annotations of terms extracted with different combinations of resources (either services or datasets) as well as how to semantically annotate the workflows themselves. Details about the semantic annotation performed appear in section 6. Table 1 shows some terms extracted by each experiment (pipeline of services) that appear in figure 1 and also the *precision* (the ratio of correctly extracted terms over all extracted terms) obtained.

Table 1. Some terms extracted for each experiment (pipeline of services) depicted in figure 1

Experiment 1	Experiment 2	Experiment 3
depression	(‘lost’,’interest’)	(‘depression’,’NN’)
insomnia	(‘daily’,’activitites’)	(‘alcohol’,’NN’)
fatigue	(‘depressive’,’episode’)	(‘antidepressants’,’NNS’)
anxiety	(‘difficulty’,’concentrating’)	(‘fatigue’,’NN’)
suicidal	(‘emotional’,’blunting’)	(‘depressive’,’JJ’)
tired	(‘eye’,’contact’)	(‘emotional’,’JJ’)
Precision = 0.23	*Precision =0.14*	*Precision =0.21*

3 Sentiment Analysis

Within psychiatric narratives there is *affective* and *informative* content. Therefore, it makes sense using lexical resources for mining affective content.

Opinion mining, also known as *Sentiment Analysis*, is a recent sub-discipline at the crossroads of information retrieval and computational linguistics which is concerned not with the topic a text is about, but with the opinion it expresses [20]. A typical approach to *Sentiment Analysis* is to start with a lexicon of *positive* and *negative* words and phrases [18], where lexical entries are tagged with their *prior polarity*, i.e. out of context taking into account only if the word(s) seem to evoke something positive or something negative. An example of such lexicons is the large valence categories from General Inquirer [19]. Some other specific resources (datasets) that have been developed are: SentiWordNet [20], and OpinionFinder’s Subjectivity Lexicon [21].

In the literature, it is possible to find approaches to *Sentiment Analysis* that range from counting the *prior polarity* of words [22] to systems that make a full compositional semantics analysis of sentence affect [23]. Nowadays it is also possible to find comparative studies [24] evaluating the effectiveness of different classifiers and showing how the use of multiple classifiers in a hybrid manner can improve the effectiveness of *Sentiment Analysis*.

To unveil the relevance of *affective* content within the psychiatric narratives selected for this study, several experiments were performed. These experiments involve using the workflow that appears on figure 2, where the *lexical resource* is made out of *positive* and *negative* words taken from the two large valence categories of General Inquirer [19]. As part of the *term detector* value-added service depicted in figure 2, a *HTML parser* is included to acquire relevant terms from URLs linked to four files (TAGNeg.html, TAGPos.html, TAGNo.html, and TAGYes.html) that can be accessed from http://www.webuse.umd.edu:9090/tags/.

From the earlier experiments with the lexicon selected, it was obvious the underlying overlap between *positive* and *negative* words. Thus, the lexicon was revised to obtain two non overlapping *sets of lexical data* tagged with their *prior*

polarity. The research study pays more attention to *negative* words, as these are more likely candidates to bring up meaningful associations between text elements and knowledge structures within psychiatric narrative. This is particularly true for the case histories selected that are about the common mental disorder of *depression*.

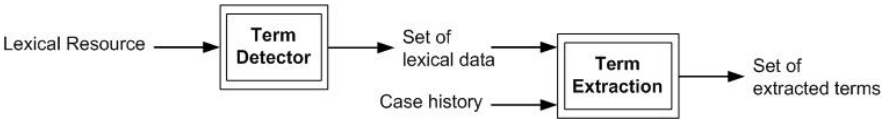


Figure 2 Data-driven workflow using two value-added services (double box)

The latest results obtained for the *Sentiment Analysis* appear in section 5. Details about the semantic annotation performed appear in section 6.

4 Semantically Meaningful Analysis

The base method for content analysis is to analyse high-frequency words and draw conclusions based on this information [25]. Therefore, content analysis implies dealing with word frequencies. Word variations ('crying', 'cry', 'cried') and synonyms can be problematic when dealing with keyword frequencies. One way to avoid this problem is to perform a *semantically meaningful analysis*, where natural language is mapped to one or more standard medical terminologies. This implies mapping synonyms and word variations to the same semantic concept, and thereby, this can be the foundation to compare clinical narratives, not being based on the actual content (the actual words), but being based on semantic concepts. This is the approach followed here. However, before going into the details of how to perform a *semantically meaningful analysis* (subsection 4.3) having as a starting point a small corpus composed of 25 online available case histories, it is worthwhile investigating first what can be achieved by re-using existing lexical resources, such as ontologies or highly specialised medical terminologies.

4.1 Using Ontologies as Lexical Resources

Using the data-driven workflow that appears in figure 2, where two value-added services (double box) have been composed, it is possible to perform experiments taking different ontologies as *lexical resources*.

As it appears highlighted in figure 2, the conversion of an ontology (lexical resource) into a *set of lexical data* implies having a *term detector*. As part of the *term detector* value-added service shown in figure 2, an *OWL parser* is included.

The OWL parser implemented detects *owl:Class*, *rdfs:subClassOf*, *ObjectProperty*, *DatatypeProperty*, and so on. After some experimentation, this research study selects as lexical resources the well known *Galen ontology* (full-galen.owl) [26] and also the *symptoms ontology* (SYMP.owl) that belongs to the Open Biomedical Ontologies (OBO) collection [27], which is widely used.

The results obtained using ontologies as lexical resources appear in section 5. Details about the semantic annotation performed appear in section 6.

4.2 Using SNOMED CT as Lexical Resource

SNOMED CT [28] terminology plays a pivotal role within HL7 Clinical Document Architecture (CDA). The ultimate goal of the current research is to aid the conversion of clinical narratives into a formal representation useful for prominent EHRs standards, and therefore, it makes sense to perform experiments that reveal the presence of standard medical terminologies within clinical narratives.

The experiments carried out follow the data-driven workflow that appears in figure 2, and make use of May 2010 release that is available for download. The file that contains the core terminology is text-based formatted. Similarly to what has been done in the above-mentioned subsection, a *Text-Based parser* is included as part of the *term detector* value-added service depicted in figure 2. The *Text-Based parser* implemented allows obtaining 39 *sets of lexical data* out of the original file (sct1_Concepts_Core_INT_20100131.txt) included in May 2010 release. Each *set of lexical data* is acquired taking into account high level abstract concepts, such as 'body structure'. Among these 39 *sets of lexical data* preference is given to two *sets of lexical data* that respectively correspond to 'finding' and 'disorder' high level abstract concepts.

Most promising results obtained using SNOMED CT as lexical resource appear in section 5. Details about the semantic annotation performed appear in section 6.

4.3 Using UMLS to enable Semantically Meaningful Analysis

The experiments performed in the previous subsections provide promising results, although word variations and synonyms do not receive much consideration. This subsection intends to use UMLS Metathesaurus [29] as a lexical resource. UMLS Metathesaurus is a very large, multi-purpose, and multilingual vocabulary database that contains information about biomedical and health related concepts, their various names, and the relationships among them [29].

The huge size of UMLS Metathesaurus makes reasonable to seek for a strategy that allows the retrieval of terms focusing on mental health, and particularly on

depression, which is the common mental disorder relevant for the case histories selected for the current research study. With this aim, this study has selected as starting point MedlinePlus [30], which provides 'depression' as a health topic within *mental health and behaviour*. Therefore, it is possible to obtain a small set of URLs linking to static Webpages from where it is feasible to acquire a *set of lexical data*. Once more, the data-driven workflow that appears in figure 2 is used. A *HTML parser* is included as part of the *term detector* value-added service that allows obtaining 95 relevant terms for *depression* from the MedlinePlus URLs selected. Initial experiments were conducted over the training set (20 case histories out of the total 25) with modest results. This was expected due to the relative small size of the *set of lexical data* obtained. However, the *set of lexical data* acquired seems to be adequate in size and representative in content. This makes it an ideal candidate to be extended by means of UMLS Metathesaurus, which is available to licensees via download, by Web interface, and an API from the UMLS Knowledge Source Server (UMLSKS) [31]. The API provides a number of functions for querying UMLS data, allowing the request of information about particular UMLS concepts, and facilitating to limit a query to a particular vocabulary or obtaining the synonyms for a particular term in a particular vocabulary.

Out of the *set of lexical data* of 95 relevant terms for *depression* from MedLinePlus (see above) and limiting the searches to SNOMED CT, it was possible to obtain 35 UMLS concepts. With the extension made, the lexical resource is labelled as *MedLinePlus Extended (see section 5)*. In UMLS a concept is a meaning that can have many different names. Each concept or meaning in the UMLS Metathesaurus has a unique and permanent concept identifier (CUI). A key goal in building the UMLS Metathesaurus is linking all the names from all the source vocabularies that mean the same (synonyms). Lexical variants, including case, spelling, and inflectional variants are considered. Thus, the 35 UMLS concepts are associated to 161 terms including word variations and synonyms.

At this point experts' advice was sought about the results obtained from the experiments performed with the training corpus and the *set of lexical data* made out of 161 relevant terms. Based on experts' feedback, the *set of lexical data* is enlarged and new queries are made, this time without limiting results to a particular vocabulary. With the new queries, a total of 91 UMLS concepts are obtained. These are associated to 625 terms that represent word variations and synonyms. Although different source vocabularies are allowed, preference is giving to SNOMED CT and also to *Thesaurus of Psychological Index Terms* [32] from the queries outputs.

Figure 3 shows part of the semantic network that can be built out of the 91 UMLS concepts, where *black links* represent direct connections while *grey links* represent indirect ones, i.e. connections that are performed by mediation of other concepts and links. As it has been highlighted, a semantic network is different from a domain ontology [33], which is a more rigid structure aimed at presenting a shared conceptualisation of a domain.

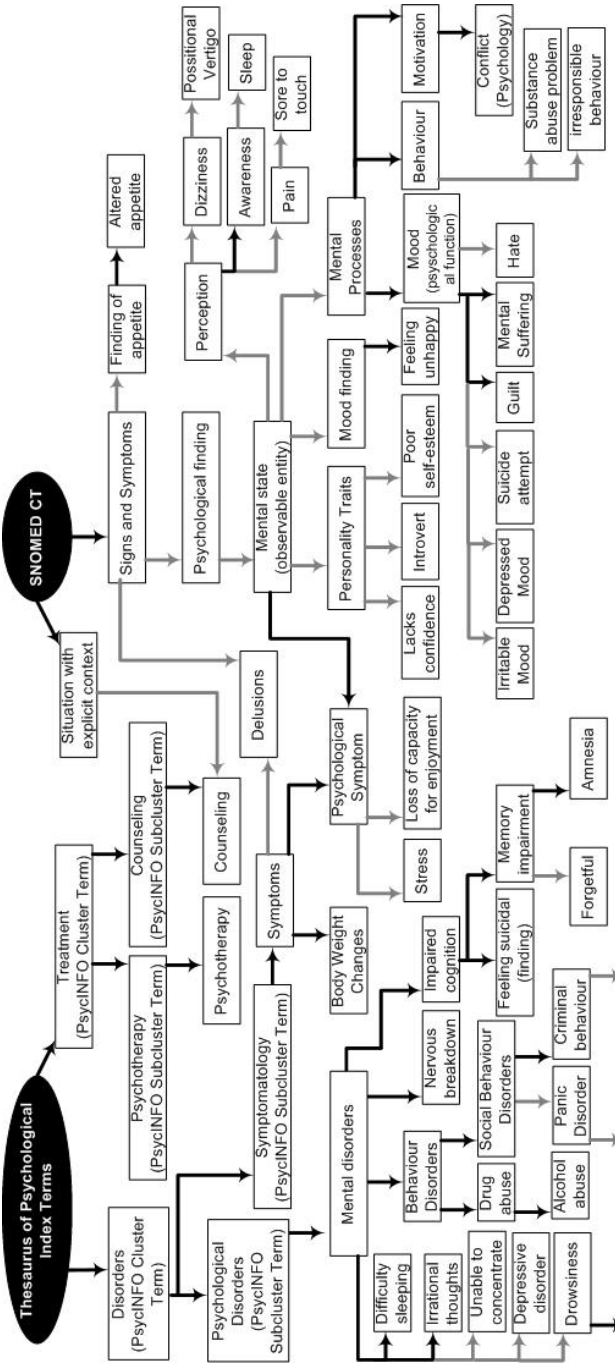


Figure 3 Part of the Semantic Network built out of the 91 UMLS concepts obtained

From a knowledge representation point of view, *semantically meaningful analysis* arises as a major asset to support building a semantic network of more than 100 interlinked UMLS concepts. Furthermore, as it is depicted in figure 3, shared concepts naturally appeared between SNOMED CT and *Thesaurus of Psychological Index Terms*, and the interconnections between these two are made clear. Hence, the semantic network obtained is quite an achievement taking into account the small size of the *training* set corpus from where it emerges.

The results obtained using MedLinePlus and UMLS Metathesaurus as a lexical resource appear in section 5. Details about the semantic annotation performed appear in section 6.

5 Content Based Analysis Results

This section summarises the performance of the eight more promising experiments made following the data-driven workflow that appears in figure 2. To measure the performance, on the one hand experts identify all the relevant terms to be extracted from the testing corpus; on the other hand the experiments that provide more promising results with the *training* corpus (20 case histories) are now repeated with the *testing* corpus (5 case histories). Table 2 shows the values obtained for three metrics: *precision*, *recall*, and *F-measure*. Their formulas appear below, where A is the correctly extracted terms; B is all extracted terms; and C is all terms to be extracted from the corpus.

Precision (P) = A/B **Recall (R)** = A/C **F-measure** = 2PR/(P+R)

Table 2. Experiments based on the data-driven workflow that appears in figure 2

N	Lexical resource	Precision %	Recall %	F-measure %
1	Valence category [negative] (Sec 3)	60	39	47
2	Galen Ontology (Subsec 4.1)	16	15	15
3	SYMP ontology (Subsec 4.1)	50	5	9
4	SNOMED CT [disorder] (Subsec 4.2)	67	3	6
5	SNOMED CT [finding] (Subsec 4.2)	38	12	18
6	MedLinePlus (Subsec 4.3)	100	4	8
7	MedLinePlus Extended (Subsec 4.3)	83	7	13
8	UMLS Metathesaurus (Subsec 4.3)	70	23	34

As it is remarked in the literature, high recall usually means sacrificing precision and vice-versa (see for example *experiment* number 6 that has 100% *precision* and very low *recall*). *F-measure* balances *recall* and *precision* in a way that gives them equal weight. From table 2, it is made plain the relevance of *affective* and *informative* content within psychiatric narratives. Experiment 1 with

a F-measure value of 47% exhibits the importance of *affective* content. Experiment 2 to 8 attempt to capture *informative* content, and among them, experiment 8 has the best F-measure with a value of 34%. The important contribution of psychiatric symptomatology to psychiatric narrative is the underlying reason that can justify the F-measure obtained for experiment number 5 using SNOMED CT [finding] as lexical resource (see subsection 4.2 for more details). Indeed, experiment number 5 has the next higher value for F-measure among the experiments performed to capture *informative* content. However, the difference in values of F-measure for experiment number 5 and number 8 exposes the benefits of using *semantically meaningful analysis* that implies mapping synonyms and word variations to the same *semantic concept* (*UMLS concept* according to experiment 8).

6 Semantic Annotation of Data and Functionality

The current research follows a service-oriented approach, where the experiments performed encourage service-reuse and *service composition* (the process of combining different services to provide a *value-added service*). A major limitation of the Web services technology is finding and composing services. A solution that appears repeatedly addressing service discovery and composition is semantic annotation, see [4][34][35]. Although technologies for semantic annotation emerge from annotation of documents, these have been more recently used to specify service functionality or to automate data-integration and service-composition tasks.

Scientific workflow systems, such as Kepler [36] or Taverna [37], have emphasised the importance of the data, and functionality does not receive full attention. This is because scientific workflows often operate on large, complex, and heterogeneous data [4]. As the current research study fosters experimentation with data-driven workflows, where lexical resources play pivotal role, the research presented here gives equal importance to *data* and *functionality*.

The current research agrees with [35] in remarking that composition of workflows of computational tasks, grid jobs, or even web services are not a new topic. Most of the proposed or developed solutions deal only with the composition of the functional part of the workflow, the data necessary to actually execute it is considered secondary [35]. In [35] semantic description of data and grid services are given by ontologies. This is also the approach followed here, although this research makes use of Web services and does not use grid services.

Figure 4 shows the domain ontology used to provide the semantic description of the data. The OWL domain ontology developed follows a modular design, where 5 ontologies have been considered: 1) the *SWRC ontology* [38] where several top level concepts and relationships have been reused, 2) the *Document Extension ontology* which is an extension of the SWRC ontology to include

patient case histories, 3) the *Mental Health and Behavior ontology* as an extension of the SWRC ontology to incorporate an adaptation of the *Mental Health and Behavior* topics from [31], 4) the *Lexical Resources ontology* as an extension of the SWRC ontology to incorporate the lexical resources used in the experiments performed (see sections 2 to 4), and 5) the *Data Set ontology* which is introduced to facilitate the linkage between inputs' and outputs' types of Web services and classes/concepts as well as instances of the other four ontologies.

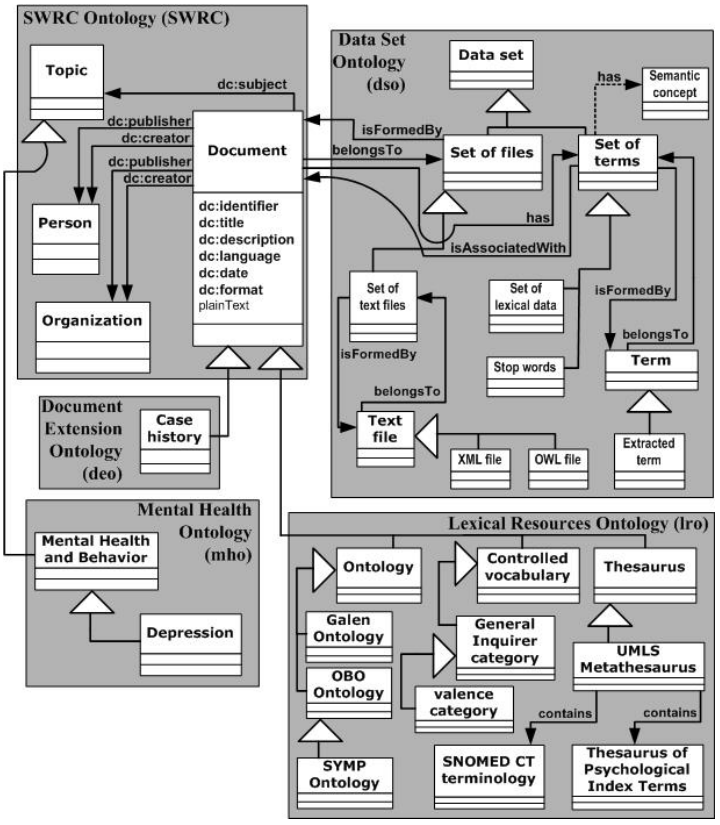


Figure 4 Overview of the domain ontology

The current research uses OWL-S, which is an OWL ontology that offers the conceptual model for semantically annotating Web services. A service in OWL-S is described by means of three elements [15]: *Service Profile*; the *Service Process Model*; and the *Services Grounding*. The current approach pays special attention to the execution graph of a *Service Process Model*, which can be composed using different types of processes (*Atomic Process, Simple Process,* and *Composite Process*) and control constructs. Figure 5 is a snapshot of OWL-S Editor plug-in

for Protégé [39] that shows the control flow and data flow for the two *composite processes* used in the data-driven workflow from figure 2.

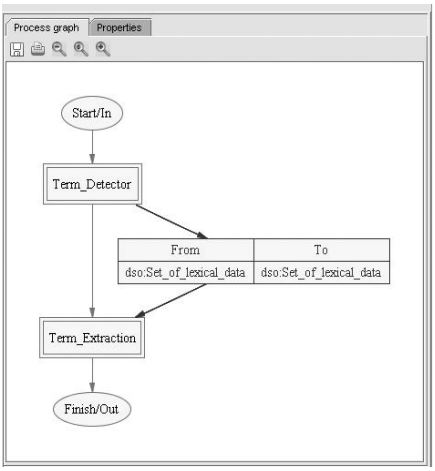


Figure 5 Control and data flow for composite processes of data-driven workflow from figure 2

Figure 6 shows an example of semantic annotation based on the domain ontology, where not only the extracted terms have been annotated, but also the UMLS concepts related to the terms extracted. Thus, the content-based analysis performed is *semantically meaningful analysis* following the data-driven workflow that appears in figure 2, and where the lexical resource is UMLS Metathesaurus (see subsection 4.3 for more details).

```
<deo:Case_history rdf:ID="c08_Exp076_185717062010">
<dc:identifier>http://priory.com/psych/obsess.htm</dc:identifier>
<dc:subject rdf:resource="http://localhost/ontology/mho#Depression"/>
<plainText rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
JY is a 50 y/o woman who presented with depression, overeating, crying and
 ... ... ...
</plainText>
<setOfLexicalData rdf:resource="http://localhost/ontology/umls#Term"/>
<termExtracted rdf:resource="http://localhost/ontology/umls_t#anxiety"/>
<termExtracted rdf:resource="http://localhost/ontology/umls_t#emotional"/>
... ... ...
<setOfLexicalData rdf:resource="http://localhost/ontology/umls#Concept"/>
<semanticConcept
rdf:resource="http://localhost/ontology/umls_cui#Distress_[C0231303]"/>
<semanticConcept
rdf:resource="http://localhost/ontology/umls_cui#Emotions_[C0013987]"/>
... ... ...
</deo:Case_history>
```

Figure 6 Semantic annotation based on the domain ontology that appears in figure 4

7 Concluding Remarks

The research presented here focuses on the challenge of finding meaningful associations between text elements and knowledge structures within psychiatric narrative. Instead of using a large corpus (millions of words) and no pre-existing knowledge; the current study uses a small corpus (5716 words) and brings into play pre-existing knowledge. The paper describes a variety of experiments for content-based analysis: *Linguistic Analysis* using NLP-oriented approaches, *Sentiment Analysis*, and *Semantically Meaningful Analysis*.

The experiments described in section 4 deal with *informative* content, while the ones described in section 3 attend to *affective* content. The performance of the more promising experiments made plain the similar relevance of *affective* and *informative* content within psychiatric narratives. *Semantically meaningful analysis* appears as the best approach (among the experiments performed) to acquire *informative* content, and from a knowledge representation point of view arises as a major asset to support building a semantic network.

Although it is not standard practice and not even recent ontology learning frameworks, such as OntoLancs or Text2Onto, bring automatic support to annotate the workflows that entail different combinations of methods, this study advocates providing automatic support to annotate functionality as well as the data. Thus, this research uses OWL and OWL-S to provide semantic description of data and services. The experiments follow a data-driven workflow, where data and functionality are equally important. In fact, most of the experiments described share the same functionality (i.e. *composite processes*), although differences in functionality appear at fine-grain level (i.e. *atomic processes*).

There is a practical value from this research that can be transmitted to legacy clinical databases that face converting clinical narratives according to prominent EHRs standards, such as HL7 CDA which favours the use of SNOMED CT.

References

1. Sharda, P., Das, A.K., Patel, V.L.: Specifying design criteria for electronic medical record interface using cognitive framework. In: AMIA annual symposium, pp. 594-598 (2003).
2. Cohen, T., Blatter, B., Patel, V.: Simulating expert clinical comprehension: Adapting latent semantic analysis to accurately extract clinical concepts from psychiatric narrative. Journal of Biomedical Informatics 41, pp. 1070-1087 (2008).
3. Doherty, G, Coyle, D., Matthews, M.: Design and evaluation guidelines for mental health technologies. Interacting with Computers 22, pp. 243-252 (2010).
4. Berkley, C., Bowers, S., Jones, M.B., Ludascher, B., Schildhauer, M., Tao, J.: Incorporating Semantics in Scientific Workflow Authoring. 17th International Conference on Scientific and Statistical Database Management. IEEE Computer Society (2005).
5. myExperiment, http://www.myexperiment.org/. Accessed May 2010.
6. EHRcom, http://www.chime.ucl.ac.uk/resources/CEN/EN13606-1/. Accessed Nov 2009.
7. openEHR Community, http://www.openehr.org/. Accessed Nov 2009.

8. HL7 CDA, http://www.hl7.org/implement/standards/cda.cfm. Accessed Nov 2009.
9. Zhou,L.: Ontology learning: state of the art and open issues. Information Technology and Management 8, pp. 241-252 (2007).
10. Gacitua, R., Sawyer, P., Rayson, P.: A flexible framework to experiment with ontology learning techniques. Knowledge-Based System 21(3), pp. 192-199 (2008).
11. Text2Onto, http://ontoware.org/projects/text2onto/. Accessed May 2010.
12. OWL, http://www.w3.org/2004/OWL/. Accessed May 2010.
13. Arguello, M., Gacitua, R., Osborne, J., Peters, S., Ekin, P., Sawyer, P.: Skeletons and Semantic Web descriptions to integrate Parallel Programming into Ontology Learning Frameworks. 11th International Conference on Computer Modelling and Simulation (2009).
14. Klein, E., Potter, S.: An ontology for NLP services. In Thierry Declerck ed., Proceedings of conference on Language Resources and Evaluation LREC'04 (2004).
15. OWL-S, http://www.w3.org/Submission/OWL-S/. Accessed May 2010.
16. Ilsey, J.E., Moffoot, A.P.R., O'Carroll, R.E.: An analysis of memory dysfunction in major depression. Journal of Affective Disorders 35, pp. 1-9 (1995).
17. Fossati, P., Guillaume, L.B., Ergis, A.M., Allilaire, J.F.: Qualitative analysis of verbal fluency in depression. Psychiatry Research 17, pp. 17-24 (2003).
18. Wilson, T., Wiebe, J., Hoffmann, P.: Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In: Conference on Human Language Technology and Empirical Methods in Natural Language Processing, pp. 347-354 (2005).
19. General Inquirer, http://www.wjh.harvard.edu/~inquirer. Accessed May 2010.
20. Esuli, A., Sebastiani, F.: SentiWordNet: A publicly available Lexical Resource for Opinion Mining. 5th Conference on Language Resources and Evaluation (LREC'06) (2006).
21. OpinionFinder's Subjectivity Lexicon, http://www.cs.pitt.edu/mpqa, Accessed May 2010.
22. Turney, P.D., Littman, M.L.: Measuring praise and criticism: Inference of semantic orientation from association. ACM Transactions on Information Systems 21(4), pp. 315-346 (2003).
23. Moilanen, K., Pulman, S.: Sentiment composition. International Conference of Recent Advances in Natural Language Processing, pp. 378-382 (2007).
24. Prabowo, R., Thelwall, M.: Sentimen Analysis: A combined approach. Journal of Infometrics 3, pp. 143-157 (2009).
25. Ryan, G.W., Bernard, H.R.: Data management and analysis methods. In: NK Denzin, YS Lincoln Eds. Handbook of Qualitative Research, Sage publications Inc, pp. 768-802 (2007).
26. Galen Ontology, http://www.co-ode.org/galen/. Accessed May 2010.
27. Open Biomedical Ontologies, http://www.obofoundry.org/. Accessed May 2010.
28. SNOMED CT, http://www.ihtsdo.org/our-standards/. Accessed May 2010.
29. UMLS Metathesaurus, http://www.nlm.nih.gov/research/umls/knowledge-sources/metathesaurus. Accessed May 2010.
30. MedLinePlus, http://www.nlm.nih.gov/medlineplus/. Accessed May 2010.
31. UMLSKS, https://login.nlm.nih.gov/cas/. Accessed May 2010.
32. Thesaurus of Psychological Index Terms, American Psychological Association, Lisa A. Gallagher ed., 10th ed. (2004).
33. Fensel, D., Horrocks, I., van Harmelen, F., McGuinness, D.L., Patel-Schneider, P.: OIL: Ontology Infrastructure to Enable the Semantic Web. IEEE Intelligent Systems 16(2), pp. 38-45 (2001).
34. Talantikite, H.N., Aissani, D., Boudjlida, N.: Semantic Annotations for web services discovery and composition, Computer Standards & Interfaces 31, pp. 1008-117 (2009).
35. Habala, O., Paralic, M., Rozinajova, V., Bartalos, P.: Semantically-Aided Data-Aware Service Workflow Composition. In: SOFSEM, NLCS 5404, pp. 317-328 (2009).
36. Kepler, https://kepler-project.org/. Accessed May 2010.
37. Taverna, http://www.taverna.org.uk/. Accessed May 2010.
38. SWRC ontology, http://ontoware.org/projects/swrc/. Accessed Nov 2009.
39. Protégé, http://protege.stanford.edu/. Accessed May 2010.

GhostWriter-2.0: Product Reviews with Case-Based Support

Derek Bridge and Paul Healy

Abstract A lot of user-generated content on the Web takes the form of records of personal experiences. Case-Based Reasoning offers a way of helping one user to reuse another's experiences from the Web. In this paper, we present GhostWriter-2.0, a Case-Based Reasoning system that supports a user who is writing a product review. GhostWriter-2.0 makes suggestions to the user, in the form of short phrases that are mined from other reviews. The purpose of the suggestions is to prompt the user to write a more comprehensive and helpful review than she might otherwise have done. We explain how GhostWriter-2.0's case base is populated with relevant and helpful reviews from Amazon. We show how it extracts and scores phrases in these reviews to decide which to suggest to the user. We report a trial with real users, in which users made greater use of GhostWriter-2.0's suggested phrases than they did of phrases suggested by a system that used a more random form of selection.

1 Introduction

Web 2.0 is the era of user-generated content. Users of the Web now produce and share content, as well as consuming it. This content takes many forms, including photos, videos, blog posts, status updates, comments, tags, and reviews.

A lot of user-generated content on the Web takes the form of records of *personal experiences* [12, 14, 13]. Epitomizing these records of personal experience are the reviews and ratings that users contribute on everything from movies to books to music to hotels to consumer goods and to on-line content, whether user-generated

Derek Bridge
Department of Computer Science, University College Cork, Ireland, e-mail: d.bridge@cs.ucc.ie

Paul Healy
Department of Computer Science, University College Cork, Ireland, e-mail: pjh1@student.cs.ucc.ie

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_35,

or otherwise. Interestingly, Web sites that solicit user reviews often solicit meta-experiences too: users can review or rate other reviews or reviewers. For example, Amazon users vote on reviews, so the site can report review helpfulness; Epinions users additionally vote on reviewers, so the site can report reviewer trustworthiness.[1] Meta-experience like this provides a partial remedy for the problem of how to ignore noisy (e.g. malicious, ill-informed or uninformative) reviews.

Once shared on the Web, one user's experiences can be reused by another user to support her in some task that she is performing. Case-Based Reasoning (CBR), often defined as reasoning from experiences [8], offers one way to support such a user. A CBR system has a case base in which it stores experiences. When CBR is used for problem-solving (e.g. classification, diagnosis, planning and design), cases often have two or three parts. The first part is a description of a previously-solved problem; the second part is a description of the solution that was used to solve the problem; and the third part, which is not always included, is an indication of the outcome of, or the numeric reward (or punishment) received when, using the recorded solution in the circumstances described by the case [8, 2]. A CBR system solves a new problem by retrieving from its case base cases that are similar to the new problem. It then adapts the solutions of the retrieved cases to the new circumstances. The solution can then be deployed, resulting in feedback (the outcome or reward). This gives the system a new experience that it can store in its case base, hence making CBR a strategy that combines problem-solving with learning [1].

Examples of using CBR to support reuse of experiences recorded on the Web include the Poolcasting system, which uses CBR to select music to play in social situations [13]; the work of [4], which uses CBR to recommend data visualizations; and CommunityCook, which extracts recipe adaptation knowledge from Web-based recipe corpora [6].

Our position is that there are in fact at least two ways in which records of experience on the Web can be reused:

- to support the user in her real-world task, e.g. booking a hotel, selecting music to play, installing software, and visualizing data; and
- to support the user when she authors new content, e.g. writing a new review.

In our own work, we have been developing the GhostWriter family of systems, which uses CBR in the second of these two ways. In GhostWriter-1.0 (see Section 6), we showed how CBR could help the user to write more comprehensive descriptions in Web-based exchange systems [16]. GhostWriter-2.0, which we describe in this paper, is our system which uses CBR to help the user to write product reviews. The GhostWriter-2.0 case base is populated with existing relevant and high quality Amazon product reviews; and the system uses the case content to assist the user to author a new review.

GhostWriter-2.0 has the potential to create a virtuous circle: if its assistance results in the user writing and submitting a better review than she otherwise would, then a new higher quality record of experience becomes available both to users mak-

[1] `www.amazon.com`, `epinions.com`

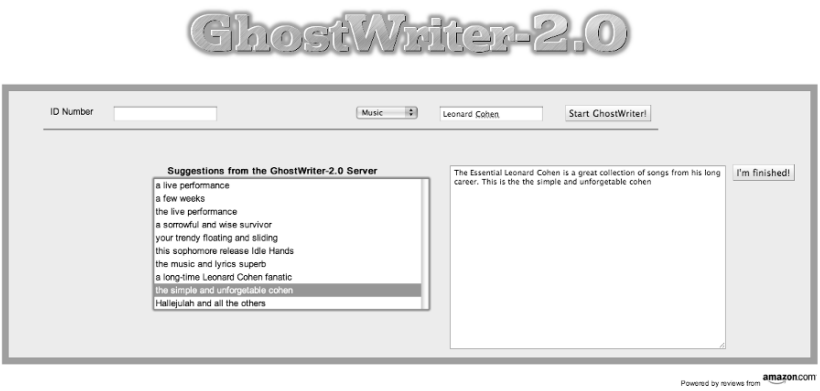


Fig. 1 A screenshot from GhostWriter-2.0

ing purchasing decisions but also to GhostWriter-2.0 next time it assists someone to write a review of a similar item.

Section 2 presents GhostWriter-2.0 from the user point of view; section 3 describes how GhostWriter-2.0 converts Amazon reviews into cases; section 4 explains how GhostWriter-2.0 populates its case base, and how it uses the cases to make suggestions; section 5 reports the results of a real user trial; and section 6 discusses related work.

2 An End-User View of GhostWriter-2.0

A user who wishes to review an Amazon product with support from GhostWriter-2.0 accesses GhostWriter-2.0 through her browser. We give a screenshot in Figure 1. The user starts at the top of the screen by using the drop-down select list to choose the category of product that she will review; in the example, the user has chosen Music. In the adjacent text-field, she enters keywords to describe the product. This might be the author of a book, the artist for a music CD, or the manufacturer of a camera, for example; in the screenshot, the user has entered "Leonard Cohen".[2]

The user next presses the *Start GhostWriter!* button. The GhostWriter-2.0 server takes the user's category and keywords and uses them to query Amazon's servers. The search returns products and their reviews. But these are not shown to the user. Instead, we use the best of these reviews to populate the case base (see Section 4.1).

The user may now type her review into the right-hand text-area. After approximately every 60 characters that the user types, GhostWriter-2.0 will make a set of suggestions, which the browser displays in the selection form to the left of the text-area. GhostWriter-2.0 selects these suggestions from the case base (see Section 4.2).

[2] The text-field in the top-left of the screenshot allows us to identify users when we are running user trials (Section 5) and is otherwise not used.

For example, in Figure 1, the user's review begins with the words "The Essential Leonard Cohen is a great collection of songs…". GhostWriter-2.0's suggestions include "a live performance" and "the simple and unforgettable cohen", this being the suggestion that the user has clicked on (see below).

A user, glancing at the suggestions, may take one of three actions:

- She might decide to include a suggestion exactly 'as is' in her review. To do this, she double-clicks on the suggestion.
- She might decide that none of the suggestions can be included verbatim, but one or more of them may inspire her to write on a related topic.
- She may find none of the suggestions helpful. In this case, she continues to type her review.

When she has finished her review, she presses the *I'm finished!* button to submit her review to the GhostWriter-2.0 server. Unfortunately, the Amazon API appears not to allow the GhostWriter-2.0 server to directly upload the review to Amazon.

3 Cases in GhostWriter-2.0

Fundamentally, cases in GhostWriter-2.0 are Amazon product reviews. We convert an Amazon product review r into a three-part GhostWriter-2.0 case $c = \langle W, S, h \rangle$ as follows. The first part, W, which can be thought of as the problem description part of the case, is the set of words that occur in the text of review r. Information about which product is being reviewed is not needed in the problem description part of the case because this contextual focus is already provided by the case base as a whole (see Section 4.1). The second part, S, which can be thought of as the problem solution part of the case, is a set of suggestions, i.e. a set of phrases that GhostWriter-2.0 might show to the author of a new review. We explain how we extract these from r in the paragraphs below. The third part of a case, h, is a non-negative integer. Recall that Amazon users can vote to signal that they found a review to be helpful. h is r's helpfulness rating. It can be thought of as the outcome or reward part of the case. It gives an indication of c's quality and, indirectly, an indication of the quality of the suggestions in S.

We have yet to explain how we extract S from r. In GhostWriter-1.0, suggestions were feature-value pairs [16]. There, we were working in the domain of Web-based exchange systems, such as classified ads systems, and so each case described an item that someone wanted to dispose of. We used regular expressions to extract the value of features such as Price (e.g. "€25 or nearest offer"), Condition (e.g. "in excellent condition"), and so on. This was adequately effective because classified ads tend to be written in a domain-specific sub-language [7] that is quite restrictive and uses a lot of recurring 'stock' phrases and phrasing.

Product reviews are written in a much less restrictive way than classified ads. Reviews of novels, for example, rarely mention the price but often give a synopsis of the plot, make comparisons with other novels by the same author or similar authors,

describe the passions aroused when reading the book, and offer opinions about the quality of the writing. The author of a new review might welcome suggestions that span all of these. Information Extraction techniques (e.g. [3]), whether based on regular expressions or something more powerful, could extract simple features such as the price, if present, or the technical attributes of digital electronics goods (e.g. their battery life). Sentiment Analysis techniques (e.g. [11]) might extract opinions and their polarity. But, even if we were to use both Information Extraction and Sentiment Analysis together, we would extract only a subset of the potentially useful suggestions.

Hence, we decided to take an approach that was less specific than either Information Extraction or Sentiment Analysis. We took the view that most of the descriptive content of a review lies in its noun phrases. They can cover: technical attributes of consumer goods; characters, places and events in books and films; authors and artists; and some of the opinion.

Noun phrases are easy to extract from text, using just quite shallow natural language processing techniques. We use OpenNLP's maximum-entropy-based 'chunking',[3] which is a form of lightweight parsing. The chunks it finds in the sentence "The entertaining and engaging characters make the book come to life" are "The entertaining and engaging characters" (noun phrase), "makes" (verb phrase), "the book" (noun phrase), "come" (verb phrase) and "life" (noun phrase). We retain only those phrases that it labels as noun phrases. In fact, we decided to retain only noun phrases that were longer than two words (because shorter noun phrases tend to lack enough descriptive content to make useful suggestions) but shorter than six words (because longer noun phrases tend to make overly specific suggestions). In the example, only "The entertaining and engaging characters" gets stored in the case as a potential suggestion. Note that there is an additional scoring process that will determine which of a case's suggestions are actually shown to the user (see Section 4.2).

In summary then, in GhostWriter-2.0 an Amazon review *r* becomes a case *c* and comprises: the set of words in *r*; the set of *r*'s noun phrases that contain three to five words; and *r*'s helpfulness rating.

4 How Ghostwriter-2.0 Works

The GhostWriter-2.0 server sits between the user's browser and Amazon's servers. The GhostWriter-2.0 server invokes operations from Amazon's Product Advertising Application Programming Interface, which is part of Amazon Web Services (AWS).[4] This API allows GhostWriter-2.0 to search Amazon's product catalog for product information and reviews.

There are two phases to GhostWriter-2.0's processing: populating the case base, and making suggestions.

[3] `opennlp.sourceforge.net`

[4] `aws.amazon.com/`

4.1 Populating the case base

As explained in Section 2, a user who wishes to write a review with support from GhostWriter-2.0 begins by entering data about the product she wishes to review: she enters its category and some keywords into her browser and submits them to the GhostWriter-2.0 server. The GhostWriter-2.0 server forwards this data to the Amazon servers. The operation it requests is an Amazon `ItemSearch` for the given category and keywords, which returns up to 4000 products in pages of up to ten products each. We additionally request that, for each product, the `ItemSearch` returns reviews, ordered by decreasing helpfulness. It returns up to five reviews per product, and it is these we use to populate the case base.

As explained in Section 3, reviews become cases. However, there is a problem that the `ItemSearch` may return duplicate reviews. This is not caused so much by customers submitting copies of their own or others' reviews, although this can happen; rather, it is because Amazon sometimes cross-posts reviews, e.g., from earlier to later editions of a book. We did not want duplicate reviews to result in duplicate cases: since suggestion scores are based in part on suggestion frequency (Section 4.2), duplicate cases would incorrectly inflate these scores. As a way of trying to avoid duplicates, we check customer ids and use only one review for each customer.

We aim to populate the case base with 250 cases. If the original `ItemSearch` fails to provide us with 250 cases (e.g. if it returns insufficient products or insufficient distinct reviews), we take again in turn each product that it does return and use an Amazon `ItemLookup` to obtain up to a further 10 reviews, again checking customer ids before inserting them into the case base.

Algorithm 1 shows how GhostWriter-2.0 populates the case base.

Note that the case base is populated afresh each time a user starts a new review. This ensures that its contents are relevant to the current product being reviewed, and that the case base takes into account the very latest data (products, reviews and helpfulness ratings) on the Amazon site. The downside is that there is an appreciable delay (up to 2 minutes) in inserting reviews into the case base. This is caused by the time it takes to launch OpenNLP and to use it to extract noun phrases from the 250 reviews. A production version of Ghostwriter-2.0 would need to be better integrated with Amazon to bring this time down.

4.2 Making suggestions

After GhostWriter-2.0 has populated its case base, the user starts typing her review into her browser. As explained in Section 2, after approximately every 60 characters that the user types, the browser uses AJAX to request a set of suggestions from the GhostWriter-2.0 server. The browser supplies the server with the current contents of the user's review. The GhostWriter-2.0 server retrieves $k_1 = 50$ cases from the case base. From the suggestions contained in these cases, it selects $k_2 = 10$ suggestions,

```
Input: product category and keywords
CB ← { };
P ← ItemSearch(category, keywords) ;   // get up to 4000 products with up
                                       // to 5 most helpful reviews each
foreach p ∈ P do
    R ← p's reviews;
    foreach r ∈ R do
        if we don't already have a review by r's customer then
            create case c from r and insert into CB;
            if |CB| = 250 then
                return CB;
foreach p ∈ P do
    R ← ItemLookup(p) ;                // get the next 10 most helpful
                                       // reviews for product p
    forall r ∈ R do
        if we don't already have a review by r's customer then
            create case c from r and insert into CB;
            if |CB| = 250 then
                return CB;
return CB;
```

Algorithm 1: GhostWriter-2.0's algorithm for populating the case base

which it returns to the user. We explain both the retrieval and the selection in more detail below.

4.2.1 Retrieval

Let the current contents of the user's review be called the *new product review* and designated *npr*.[5] This is the set of words that the user has typed. GhostWriter-2.0 retrieves from the case base the k_1 cases that are most similar to the *npr*. We measure similarity using the Jaccard similarity coefficient:

$$sim(npr, c = \langle W, S, h \rangle) = \frac{|npr \cap W|}{|npr \cup W|}$$

4.2.2 Suggestion selection

At this point, GhostWriter-2.0 has retrieved k_1 cases C from the case base, and in each case $c = \langle W, S, h \rangle$ there is a set of suggestions S; each suggestion is a noun phrase. GhostWriter-2.0 must select the k_2 suggestions that it will send back to the

[5] We avoid the word "query", which is more common in CBR, since we have found it leads to confusion.

browser for display. When considering a unique candidate suggestion, $s \in \bigcup_{\langle W,S,h\rangle \in C} S$, several criteria seem relevant, discussed over the next few paragraphs.

Consider the number of retrieved cases in which a suggestion s appears:

$$freq(s) = |\{\langle W,S,h\rangle \in C : s \in S\}|$$

To some extent, the more frequent s is, the more its content is something that different reviewers want to talk about, and this makes it a better suggestion. However, frequency favours short suggestions as these are the ones that are more likely to recur. The downside of this is that short, recurring noun phrases are more likely to be vague or generic.

Suppose instead we consider the length in words of the suggestions, $length(s)$. To some extent, the longer s is, the more descriptive it is, and this makes it a better suggestion. But length alone may favour overly specific suggestions.

We found that the product of frequency and length offered a good balance between the two criteria:

$$score(s) = freq(s) \times length(s)$$

The short, three-word phrase "a fantastic album" needs to appear in two separate cases ($2 \times 3 = 6$) if it is to have a score that betters that of the five-word phrase "an excellent, lively rock classic", assuming this appears in only one case ($1 \times 5 = 5$).

Many suggestions share the same score. To break ties, we sum and compare the helpfulness of the reviews that contain the two suggestions. Formally, if $score(s) = score(s')$, then s will be ranked higher than s' if

$$\sum_{h_s \in \{h:\langle W,S,h\rangle \in C \wedge s \in S\}} h_s \quad > \quad \sum_{h_{s'} \in \{h':\langle W',S',h'\rangle \in C \wedge s' \in S'\}} h_{s'}$$

(If s and s' have the same total helpfulness, then tie-breaking is arbitrary.)

In addition to this scoring, we also want to apply other criteria.

We do not want to make a suggestion s if it is already a noun phrase in the user's *npr*. This is quite simple-minded. In the future, it may be worth considering ways of measuring the similarity of semantic content: we would discard s if the *npr*'s content was similar enough to cover the content of s.

We also do not want to make a suggestion s if it is one that we have made several times already. This criterion is not one that we built into an early version of GhostWriter-2.0. But in a small-scale pilot study that we conducted prior to the user trial reported in Section 5, we found that users preferred versions of the system that more often made fresh suggestions over versions that allowed suggestions to linger. We subsequently decided to limit the number of times a suggestion could be made. Specifically, if s has already been suggested $\theta = 4$ times, then it cannot be suggested again, allowing another suggestion to take its place. This increases the number of different suggestions that get made, but the suggestion turnover is not so great as to be counter-productive or distracting to the user.

Input: the case base, *CB*, and the user-supplied new product review, *npr*
forall $c \in CB$ **do**
  compute $sim(npr, c)$;
$C \leftarrow$ the k_1 most similar cases from *CB*;
$Candidates \leftarrow \bigcup_{\langle W,S,h\rangle \in C} S$;
remove from *Candidates* any *s* that is a noun phrase in the *npr*;
remove from *Candidates* any *s* that has been suggested to this user more than θ times;
forall $s \in Candidates$ **do**
  compute $score(s)$;
return *the* k_2 *highest scoring suggestions from Candidates using helpfulness ratings for tie-breaking*;

Algorithm 2: GhostWriter-2.0's algorithm for making suggestions

Algorithm 2 shows how GhostWriter-2.0 makes suggestions.

5 Experimental Evaluation

Here, we report the results of a user trial. For comparison purposes, we developed a version of GhostWriter-2.0 that made less intelligent use of the cases. It populates its case base in the same way as GhostWriter-2.0 (Section 4.1). So we know it has a set of relevant and helpful cases. But instead of retrieving the $k_1 = 50$ cases that are most similar to the *npr*, it retrieves $k_1 = 50$ cases at random from the case base. Like GhostWriter-2.0, it will not make suggestions that are already contained in the *npr*, nor will it make a suggestion more than $\theta = 4$ times. But otherwise, the $k_2 = 10$ suggestions that it makes are drawn at random from the retrieved cases: it does not use the *score* function, nor the helpfulness ratings. In this section we will refer to these two systems as GW-2.0 (for GhostWriter-2.0) and GW-R (for the more random version).

The two systems were evaluated by twenty users. It turned out that none of our volunteer users had written Amazon reviews before, but the vast majority had read Amazon reviews when making purchasing decisions. In our trial, each user reviewed two products with which they were familiar, either two music CDs or two books of their own choosing. As they wrote their reviews, they received suggestions from one of the GhostWriter systems. Ten users had help from GW-2.0 for their first review, and help from GW-R for their second review. The other ten users had help from GW-R first, and then GW-2.0. They were unaware of the difference between the systems.

During the experiment, GW-2.0 and GW-R recorded: the review the user typed; the time taken to write the review; and the suggestions that were explicitly incorporated, i.e. when the user double-clicked on a suggestion. We also administered a questionnaire to each participant.

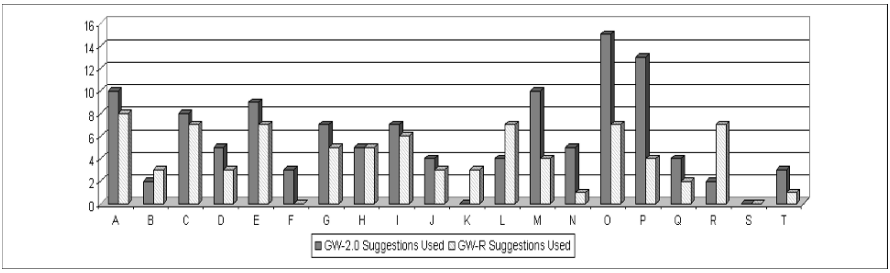


Fig. 2 The number of suggestions that users directly incorporated into reviews

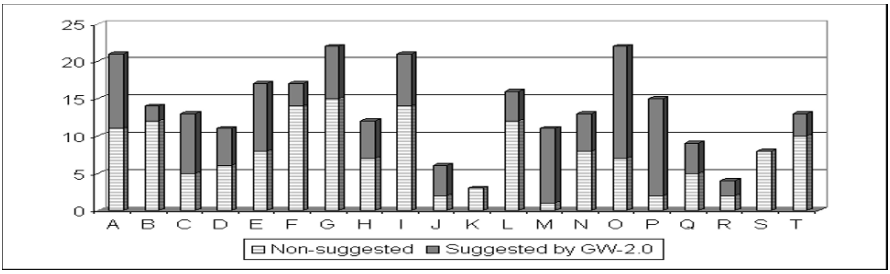


Fig. 3 Noun phrases in reviews written with GW-2.0

At one level, reviews written with support from GW-2.0 and with support from GW-R were similar. Their average length was a little over 150 words in both cases; users created their reviews at an average of 13–14 words per minute in both cases; and the average number of descriptive noun phrases in the reviews (which we took to be noun phrases of more than two words) was also 13–14 in both cases. But what is interesting is the breakdown of those noun phrases.

Figure 2 shows, for each user (designated A to T), how many suggestions they directly used (by double-clicking on them). In total, 116 GW-2.0 suggestions were directly incorporated, an average of 5.8 per user, compared with only 83 GW-R an average of 4.15. Fourteen of the twenty users used more GW-2.0 suggestions than GW-R ones; one user used the same number of suggestions from both; and one used none at all from either system. We take this as an indication that GW-2.0's greater reuse of Web-based experience data (case similarity, frequency across reviews, and helpfulness) promotes more useful suggestions.

Figure 2 gives no indication of how many other descriptive noun phrases there are in user reviews. Figure 3 shows this in the case of GW-2.0. It enables us to see that, while users vary, directly incorporated suggestions account on average for about 43% of the descriptive noun phrases in these user reviews (116 out of the 268 noun phrases of more than two words). We do not have space to show the graph for GW-R but on average only about 30% of descriptive noun phrases were ones GW-R suggested (83 out of 275 noun phrases of more than two words).

Figures 4, 5 and 6 summarize results from the questionnaire. Nineteen of the twenty people agreed or strongly agreed that GW-2.0 helped them to write a com-

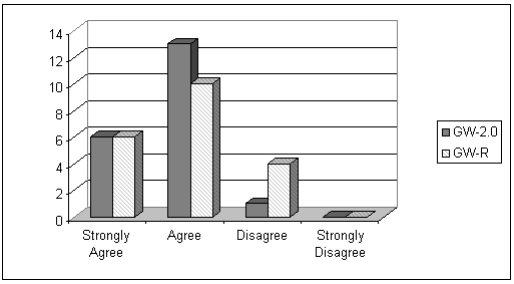


Fig. 4 "The system helped me to write a comprehensive review"

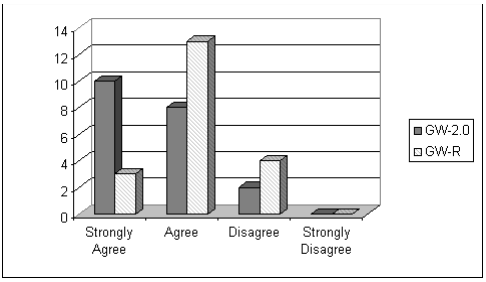


Fig. 5 "I found the suggestions helpful"

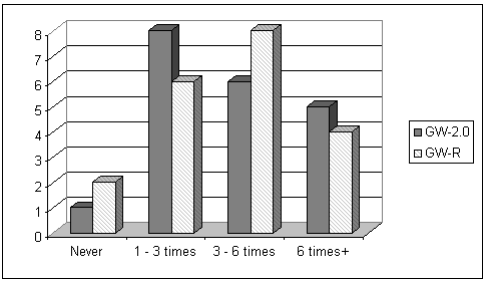


Fig. 6 "The number of times I was inspired by suggestions but I didn't click on them"

prehensive review; for GW-R, sixteen people agreed or strongly agreed (Figure 4). Eighteen people agreed or strongly agreed that GW-2.0's suggestions were helpful, more than half of them strongly agreeing; for GW-R, sixteen people agreed or strongly agreed, only a minority strongly agreeing (Figure 5). We also asked participants to estimate for each review that they wrote how many times they were inspired by a suggestion but did not actually double-click on it and so did not incorporate it directly. This was obviously very subjective and not likely to be reliable. But it at least gives an impression of the extent to which the numbers reported in Figures 2 and 3 understate the helpfulness of the systems. According to the responses to this (Figure 6) GW-2.0 and GW-R were fairly evenly-matched in their ability to inspire, and inspired their users one or more times in all but three reviews.

6 Related Research

Systems, like GhostWriter-2.0, that make suggestions to a user who is typing text into her browser now see widespread use across the Web. Google Suggest is one example.[6] It proposes a set of search queries that other users have issued and that are similar to (e.g. completions of) the query that the user is currently entering. It derives its value from the vast search experience that is implicitly captured by Google's logs of previous searches.

There has been an amount of academic research into tasks such as phrase prediction and sentence completion, especially to support email composition. This can be useful in administrative and call centre environments, where sentences often recur across emails. The research tends to focus on data structures for fast prefix matching. For example, Grabski & Scheffer propose an *inverse index structure* to retrieve from a corpus sentences that could complete the sentence that the user is typing [5]. Similarly, Nandi & Jagadish propose what they call *fussy trees* to support phrase prediction [10]. On the one hand, there is no doubt that GhostWriter-2.0's performance could be enhanced if reviews were indexed in a way that was optimized for making suggestions; this is not possible while we are confined to accessing the reviews through the existing Amazon API. On the other hand, it is important to emphasize that GhostWriter-2.0 is not engaged in content prediction or completion. GhostWriter-2.0 prompts the user to write a more helpful review. A GhostWriter-2.0 suggestion that is not directly incorporated but which provokes the user into including helpful content that she otherwise would not have thought to include, even if it disagrees with the content of the suggestion, is still a successful one.

Lamontagne & Lapalme use Case-Based Reasoning techniques for the task of generating email replies, not just completing sentences within replies [9]. Cases in their case base are request-response pairs. The user presents to the system an email request that she has received. The system retrieves a case that deals with a similar request. It uses heuristics to annotate the response to indicate sentences that are optional, which the user might delete, and phrases that are likely to be specific to the original response (e.g. names and dates), which the user might modify. This is ambitious work but again its focus is different from ours. The assumption in their work is that the domain is one where 'stock responses' are common.

Finally, we should mention GhostWriter (which we will here refer to as GhostWriter-1.0), the predecessor to GhostWriter-2.0 [16]. Inevitably, the systems have many similarities, but they are also very different too. First, their domain of application is quite different. We designed GhostWriter-1.0 to support the users of Web-based exchange systems, such as classified ads systems, where the user is usually describing an item, such as a cot, bicycle or wardrobe, that she wants to dispose of. Second, GhostWriter-1.0 can use Information Extraction technology to extract suggestions from cases because, in its domain, comprehensive textual descriptions can be thought of as containing the value of features, such as the condition of the

[6] `www.google.com/support/websearch/bin/answer.py?hl=en&answer=106230`

item, its price, delivery terms, and so on. Third, in GhostWriter-1.0 one item has one description, whereas in GhostWriter-2.0 one product may have several reviews. Fourth, GhostWriter-1.0's item descriptions do not come with helpfulness ratings. Rather, if an item is successfully disposed of through the exchange system, then we regard its description as having been a successful one, and make it available to the CBR system. All of these differences led us to a rather different scoring system for use in GhostWriter-1.0. Fifth, unlike GhostWriter-2.0, GhostWriter-1.0 does not populate its case base afresh for each use of the system. Rather, it has a single, system-wide case base. Finally, GhostWriter-1.0 was only evaluated off-line, using simulated users. However, its deployment in a real Web site is to begin soon.

7 Conclusions and Future Work

We have presented GhostWriter-2.0, which reuses the experience captured in Amazon product reviews and their meta-review data to make suggestions to a user who is writing a new product review. Our user trial has very promising results, showing a high level of use of the suggestions, both directly and indirectly. In the trial, GhostWriter-2.0 was more helpful to users than a less sophisticated version with an element of randomness to its selections. The differences, however, were small. This may be because, irrespective of which system was being used, users in the trial relied on the suggestions to a degree greater than they would in reality, where they would be writing reviews for endogenous reasons, rather than at our behest.

There are many lines of future research. First, we note that in research into recommender systems, there is a concern to ensure that each set of recommendations is diverse [15]. At this stage, we do not know whether users regard GhostWriter-2.0's suggestions as diverse. This is something we would like to measure and, if necessary, to explicitly enhance using the kinds of techniques developed in recommender systems research. Second, we would like to investigate ways to reduce the time it takes to populate the case base. One approach is closer integration between GhostWriter-2.0 and the on-line store that it supports than is possible through the Amazon API. An alternative, where closer integration is not possible, is to populate the case base in the background, while the user is entering the first sentence or so of her review. Third, we would like to make the system less cumbersome to use. Instead of asking the user for information about the product that she wants to review (its category and keywords), it should be possible to extract this information from the page the user is visiting. However, in preliminary work on this, we have already found that there is a balance to be struck. We tried using the product title, and we found that this was so specific that the case base contained reviews from too narrow a set of past reviews. On the other hand, we tried using tags that users had assigned to products. This time we had two problems: many products had no or very few tags; and tags were often so general that the case base contained reviews that were not adequately relevant. An exciting possibility would be to use the recommender system techniques that sites such as Amazon use when recommending related products.

Fourth, we would like to evaluate GhostWriter-2.0 in a real setting. For example, if it were possible to upload reviews from GhostWriter-2.0 to Amazon, we could measure whether these reviews accrue higher helpfulness ratings than reviews authored without GhostWriter-2.0. Finally, we are keen to extend the GhostWriter family into new domains where authors can benefit from the kind of support that GhostWriter systems can offer.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications **7**(1), 39–59 (1994)
2. Bridge, D.: The virtue of reward: Performance, reinforcement and discovery in case-based reasoning. In: H. Muñoz-Avila, F. Ricci (eds.) Procs. of the 6th International Conference on Case-Based Reasoning, LNAI 3620, p. 1 (2005)
3. Etzioni, O., Banko, M., Soderland, S., Weld, D.S.: Open information extraction from the web. Communications of the ACM **51**(12), 68–74 (2008)
4. Freyne, J., Smyth, B.: Many Cases Make Light Work for Visualization in Many Eyes. In: D. Bridge, et al. (eds.) Procs. of WebCBR: The Workshop on Reasoning from Experiences on the Web (at the 8th ICCBR), pp. 25–44 (2009)
5. Grabski, K., Scheffer, T.: Sentence completion. In: M. Sanderson, et al. (eds.) Procs. of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 433–439. ACM Press (2004)
6. Ihle, N., Hant, A., Althoff, K.D.: Extraction of Adaptation Knowledge from Internet Communities. In: D. Bridge, et al. (eds.) Procs. of WebCBR: The Workshop on Reasoning from Experiences on the Web (at the 8th ICCBR), pp. 35–44 (2009)
7. Kittredge, R., Lehrberger, J.: Sublanguage: studies of language in restricted semantic domains. de Gruyter (1982)
8. Kolodner, J.L.: Case-Based Reasoning. Morgan Kaufmann (1993)
9. Lamontagne, L., Lapalme, G.: Textual reuse for email response. In: P. Funk, P.A.G. Calero (eds.) Procs. of the 7th European Conference on Case-Based Reasoning, LNCS 3155, pp. 234–246. Springer-Verlag (2004)
10. Nandi, A., Jagadish, H.V.: Effective phrase prediction. In: C. Koch, et al. (eds.) Procs. of the 33rd International Conference on Very Large Data Bases, pp. 219–230. ACM Press (2007)
11. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval **2**(1-2), 1–135 (2008)
12. Plaza, E.: Semantics and Experience in the Future Web. In: K.D. Althoff, et al. (eds.) Procs. of the 9th European Conference on Case-Based Reasoning, LNCS 5239, pp. 44–58. Springer Verlag (2008)
13. Plaza, E., Baccigalupo, C.: Principle and Praxis in the Experience Web: A Case Study in Social Music. In: D. Bridge, et al. (eds.) Procs. of WebCBR: The Workshop on Reasoning from Experiences on the Web (at the 8th ICCBR), pp. 55–63 (2009)
14. Smyth, B., Champin, P.A.: The Experience Web: A Case-Based Reasoning Perspective. In: S. Craw, et al. (eds.) Procs. of the Workshop on Grand Challenges for Reasoning From Experiences (at the 21st IJCAI), pp. 53–61 (2009)
15. Smyth, B., McClave, P.: Similarity vs. diversity. In: D.W. Aha, I. Watson (eds.) Procs. of the 4th International Conference on Case-Based Reasoning, LNCS 2080, pp. 347–361. Springer (2001)
16. Waugh, A., Bridge, D.: An Evaluation of the GhostWriter System for Case-Based Content Suggestions. In: L. Coyle, et al. (eds.) Procs. of the 20th Irish Conference on Artificial Intelligence and Cognitive Science, pp. 264–273 (2009)

SHORT PAPERS

Dynamic Programming Algorithm vs. Genetic Algorithm: Which is Faster?

Dušan Petković[1]

Abstract The article compares two different approaches for the optimization problem of large join queries (LJQs). Almost all commercial database systems use a form of the dynamic programming algorithm to solve the ordering of join operations for large join queries, i.e. joins with more than dozen join operations. The property of the dynamic programming algorithm is that the execution time increases significantly in the case, where the number of join operations in a query is large. Genetic algorithms (GAs), as a data mining technique**,** have been shown as a promising technique in solving the ordering of join operations in LJQs. Using the existing implementation of GA, we compare the dynamic programming algorithm implemented in commercial database systems with the corresponding GA module. Our results show that the use of a genetic algorithm is a better solution for optimization of large join queries, i.e., that such a technique outperforms the implementations of the dynamic programming algorithm in conventional query optimization components for very large join queries.

1 Introduction

In a relational database system (RDBMS), a good and effective query evaluation is essential for efficient processing of queries. The evaluation of a query has two components: the query optimizer and the query execution engine [3]. The query optimizer takes an internal representation of a relational query as input and generates an "optimal" query execution plan (QEP) for the given query.

The query optimization process comprises several phases and is described in [5]. The strategy, which is usually used in commercial DBMSs to search for the optimal execution plan is called dynamic programming algorithm. In relation to join operations, this algorithm applies the exhaustive search strategy.

A genetic algorithm is an iterative process [4], in which the existing chromosomes are rearranged each time to get an optimal solution. The decision about the quality of each solution is calculated using the fitness function.

1 University of Applied Sciences, Rosenheim, Germany
petkovic@fh-rosenheim.de

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_36,

Genetic algorithms, as all evolutionary algorithms comprise always three different phases. The first one, initialization, specifies the way, how the member of population are coded. The second phase is the evolution cycle. This cycle simulates the genesis of the new chromosomes and the disappearance of the old ones. In this phase the population is modified using the following operators: selection, recombination and mutation. The last phase is the end criterion. Selecting the fittest chromosomes in several iteration steps makes the population always better. The end criterion, which is specified already in the first phase, stops the whole process and produces the final result.

The article compares the different approaches for the optimization of large join queries and shows that the genetic algorithm is a better solution for the ordering problem for joins in large join queries than the implementation of the dynamic programming algorithm in commercial database management systems, in the case of relational queries with more than 24 join operations.

1.1 Related Work

The problem of ordering of join operations has been discussed in several papers. The first implementation of this problem is due to System R [1]. The main disadvantage of the algorithm used for ordering of join operations in System R is that it needs to process a very large number of QEPs during its execution. All existing DBMSs today use some form of the algorithm implemented in System R.

The comparison between genetic programming and GAs has been published in [7]. The experimental part of this paper shows that genetic programming works better for the optimization problem of large join queries than GAs. Another paper, which examines genetic programming, is [13]. The conclusion of this paper, where genetic programming is compared to another group of the algorithms called randomized algorithms ([6] and [14]) is that the latter performs better.

The paper [2] introduces genetic algorithms as a way to solve the ordering problem of large join queries. In this paper genetic algorithms are compared with the corresponding component of System R for relational queries up to 16 join operations. The experimental part of this paper shows that GA can find a better execution plan than the corresponding component of System R.

Until now, two commercial database systems have implemented a query optimization component, which uses GAs for solving the ordering problem. The query optimization module of the PostgreSQL database system, which uses GA to order tables in large join queries is called GEQO [12]. The comparison between the both PostgreSQL components with the equivalent functionality has been published in [11]. The other implementation concerns IBM DB2 [8, 9].

The rest of the paper is organized as follows. Section 2 gives a description of the implementation of the genetic algorithm in the PostgreSQL database system. The experimental evaluations are shown in Section 3. First, we run queries with

up to 32 join operations using two different systems: MS SQL Server and PostgreSQL, with its GEQO module activated. After that, we compare their execution times of the both system. In the second part of this section we investigate the both query optimization components of the PostreSQL system: the one, which uses the dynamic programming algorithm to execute large join queries and the other, which uses a genetic algorithm for this task. (The test bed used here is equivalent to the one used in the first part of the section.) Section 4 gives conclusions of the paper.

2 Implementation of GA in PostgreSQL

The particular genetic algorithm used for the GEQO module had been developed and implemented at Colorado State University. The leader of the project called GENITOR (GENetic ImplemenTOR) was Darrell Whittley. The GENITOR algorithm has been successfully used in solving different optimization problems. The advantage of this algorithm is that it produces one new chromosome at a time, so inserting a single new individual is simple. Another advantage is that the insertion operation automatically ranks the individual relative to the existing ones. For this reason, no further measures of relative fitness are needed [15].

The implementation of the GEQO module is based upon the travelling salesman problem. All possible execution plans are coded in the form of integer strings. In other words, a string represents the join ordering of a relation to the other relations in the given query. For each join sequence considered, the cost of performing the query is estimated. Join sequences with the least fit chromosomes are replaced by the new ones. (This allows the system to use the smaller number of iterations to reach the goal.) The new candidates are generated by randomly combining genes of the "best" existing chromosomes. This process is repeated until a predefined number of join sequences have been considered. After that, the best join sequence is used to generate the QEP.

3 Experimental Evaluations

In the experimental evaluation we use the following database systems: two with the conventional query optimization component (MS SQL Server and PostgreSQL) and the GEQO module of the PostgreSQL system. All systems are installed directly on a desktop computer with the MS Windows XP SP3. The computer uses the AMD Athlon 64 X2 Dual Core processor with the clock rate of 2*2.00 GHz and with 3 GB RAM. For our experiments, we use the database with more than 30 tables. For each query optimization component we run 32 different

queries with up to 32 join operations. To produce the average execution time, each query has been executed 5 times with each system and the average of the times has been calculated.

3.1 Evaluation with MS SQL Server

For this evaluation, we used SQL Server 2008 Enterprise Edition [10]. As can be seen in Figure 1, the query optimization component of SQL Server shows better performance than the PostgreSQL GEQO module for queries with less than 10 join operations. After that, the execution times for the both systems are approximately equal for all queries with less than 25 joins. Starting with the query with 25 joins, the execution time of the SQL Server system grows rapidly, while the execution time of the GEQO module remains about linear.

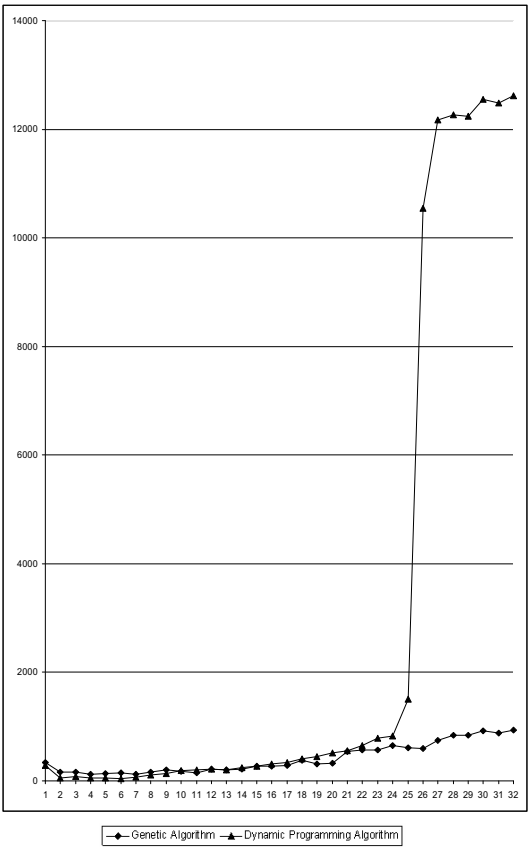


Figure 1: GEQO module vs. SQL Server query optimization component

3.2 Comparison of GEQO Module with the PostgreSQL Dynamic Programming Component

Each user of the PostgreSQL database system can optionally use either the GEQO module or the exhaustive search component for the execution of join queries. To disable, i.e., enable the GEQO module, the **geqo**() system parameter is used. This parameter can be set in the configuration file of the PostgreSQL system.

The results of the comparison of the two different PostgreSQL components, which order join operations in large join queries are similar to the results from the last subsection. The default component of the PostgreSQL system starts to perform worse than the GEQO module starting with the queries with more than 14 join operations. Figure 2 shows the comparison of these two components.

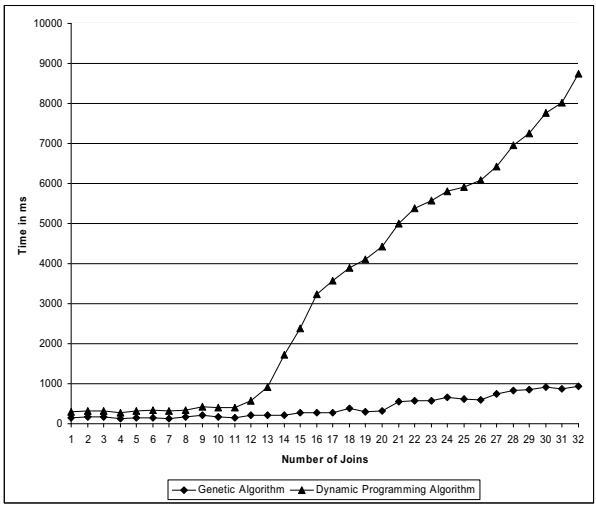


Figure 2: GEQO module vs. PostgreSQL dynamic programming algorithm

4 Conclusions

As the evaluation section of the paper shows, the use of genetic algorithms for the ordering problem of join operations in LJQs can be recommended. For this reason, it is obvious that IBM develops and implements such an algorithm for DB2 UDB. In relation to this module, it can be expected that IBM will implement several significant improvements to the CGO component. After that, in one of the future versions of DB2, the CGO could be added to the entire system as an extender. (The CGO module should be tested together with advanced features in DB2, to get the complete picture of the viability of the module [9]).

The implementation of the GEQO module supports only the left-deep tree query processing. The left-deep tree query processing means that all internal nodes of a query tree have at least one leaf as a child and inner nodes of all join operations are always database relations. For this reason, the query optimizer of the PostgreSQL system would benefit from extensions of the GEQO module, where the more general representation of query trees called bushy-tree is supported. The reason for this is that the bushy tree query processing is more preferable than left-deep tree query processing in the case of the hash join technique.

Besides the existing implementation of the GEQO module in PostgreSQL and the efforts for the implementation of CGO for DB2, we are not aware of any other existing implementation of GAs or other algorithms for solving the optimization problem of large join queries. As the results of this paper show, it is obvious that the further research in applying and implementing of GA as a component of query optimizers for other commercial RDBMSs, such as Oracle and MS SQL Server, is recommended.

References

1 Astrahan, M.M. et al. – Access Path Selection in a Relational Database Management System, in Proc. of the ACM SIGMOD Conf. on Management of Data, Boston, June 1979, pp.23-34.
2 Bennett, K.; Ferris, M. C.; Ioannidis, Y. - A genetic algorithm for database query optimization, Tech. Report TR1004, Univ. Wisconsin, Madison, 1991
3 Chaudhuri, S. – An Overview of Query Optimization in Relational Database Systems, Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, Seattle, 1998.
4 Goldberg, D. E. – Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, 1989.
5 Ioannidis, E. - Query Optimization,http://infolab.stanford.edu/~widom/cs346/ioannidis.pdf.
6 Ioannidis, Y. E.; Kang, Y. C. - Randomized Algorithms for Optimizing Large Join Queries, Proc. of the 1990 ACM-SIGMOD Conference on the Management of Data. Atlantic City, NJ, 1990, pp. 312-321.
7 Lahiri, T. Genetic Optimization Techniques for Large Join Queries, in Proc. of the 3rd Genetic Programming Conf., 1998, pp.535-40.
8 Muntes-Mulero, V.; Aguilar-Saborit, J.; Zuzarte, C; Larriba-Pey, J. – SGO: A Sound Genetic Optimizer for Cyclic Query Graphs, in Alexandrov, V.N. et al. Proceedings of ICCS 2006, Part I, LNCS 3991, 2006, pp. 156-163.
9 Muntes-Mulero, V. et al. – Analyzing the Genetic Operations of an Evolutionary Query Optimizer, in Bell, D. and Hong. J. (Eds.), Proc. of BNCOD 2006, LNCS 4042, 2006.
10 Petković, D. – SQL Server 2008, A Beginner's Guide, McGraw Hill, 2008.
11 Petković, D. - Comparison of Different Solutions for Solving the Optimization Problem of Large Join Queries, The Second Int. Conf. on Advances in Databases, Knowledge, and Data Applications, Les Menuires, France, 10.4.-14.4.2010.
12 PostgreSQL, http://www.postgresql.org
13 Stilger, M., Spiliopoulou, M. – Genetic Programming in Database Query Optimization, in Proc. of the 1st Genetic Programming Conference, 1996, pp. 388-93.
14 Steinbrunn, M., Moerkotte, G., Kemper, A. - Heuristic and randomized optimization for the join ordering problem. VLDB Journal, 6, 3 (Aug. 1997), Springer, New York, pp. 191-208
15 Whittley, D. – An Overview of Evolutionary Algorithms, Journal of Information and Software Technology 43, pp. 817-31, 2001.

Automatic Detection of Pectoral Muscle with the Maximum Intensity Change Algorithm

Zhiyong Zhang[1], Joan Lu, Yau Jim Yip

Abstract The accurate segmentation of pectoral muscle in mammograms is necessary to detect breast abnormalities in computer-aided diagnosis (CAD) of breast cancer. Based on morphological characteristics of pectoral muscle, a corner detector and the Maximum Intensity Change (MIC) algorithm were proposed in this research to detect the edge of pectoral muscle. The initial result shows that the proposed approach detected pectoral muscle with high quality.

1 Introduction

The pectoral muscle is commonly seen in the *medio-lateral oblique* (MLO) view of mammograms. The area of pectoral muscle in a mammogram is approximately a triangular region locating on the upper position of a breast. It is not easy to separate pectoral muscle from breast areas because the superimposition of pectoral muscle and breast tissue in mammograms.

Karssemmeijer [4] assumed that the edge of pectoral muscle is nearly a straight line and adopted the Hough transform and some threshold values to detect the area of pectoral muscle. With the geometric and anatomical information, Ferrari et al[5] improved Karssemmeijer's method with Gabor wavelet filter bank to segment pectoral muscle. Kwok [6] proposed a adaptive algorithm that uses straight line estimation to detect the edge of the pectoral muscle with information of position and shape of the pectoral muscle, then improve with iterative cliff detection. In addition, Mustra et al. [7] adopted bit depth reduction, dyadic wavelet decomposition and the Soble edge filter to detect the edge of pectoral muscle. Raba et al. [8] employed the curvature to detect the left and right side of breast profile to determine the orientation of breast area, and used region growing to segment the area of pectoral muscle. Based on the graph theory, Ma et al. [9] used the adaptive pyramids (AP) and minimum spanning trees (MST) to detect the initial area of pectoral muscle, then adopted the active contour to refine the initial area. Recently,

[1] University of Huddersfield, UK
z.zhang@hud.ac.uk

M. Bramer et al. (eds.), *Research and Development in Intelligent Systems XXVII*,
DOI 10.1007/978-0-85729-130-1_37,

Zhou et al. [10] adopted gradient-based directional kernel (GDK) to identify pectoral muscle.

2 Pectoral Muscle Detection

The key challenge for pectoral muscle segmentation is how to overcome the superimposition of pectoral muscle and breast tissue in mammograms. Based on morphological characteristics of pectoral muscle, the proposed approach includes three steps to complete the task of pectoral muscle detection. Figure 1 shows the schema of the proposed approach. To detect pectoral muscle from mammograms, it is necessary to segment breast area first. In this research, an automated breast segmentation (ABS) approach [12] was adopted to segment breast areas and get the boundary of them. To detect the orientation of a breast, a simple method [13] was used to detect the orientation of breast

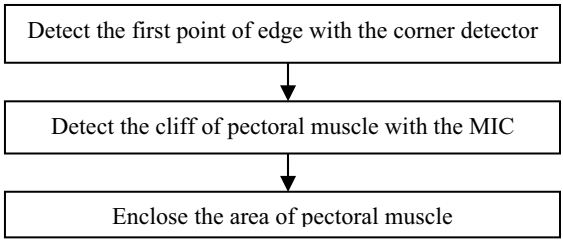


Figure 1: the schema of the proposed approach

2.1 Detect the first point of edge of pectoral muscle

Base on morphological characteristics of pectoral muscle, a corner detector in this research was developed to detect the first point (E_1). The corner detector (see figure 2) that consists of three 3x3 pixels local areas, GH, PM and GV, were moved to detect the first point (E_1) from the pectoral muscle side to breast skin-line side at the top part of the segmented breast area. A candidate point is considered as the first point (E_{1d}) when M_{HD} is maximum and M_{VD} is more than 5. The figure 2(a) shows the corner detector for detecting the first point of edge of pectoral muscle of the left breast, and the figure 2(b) shows the corner detector for detecting the first point of edge of pectoral muscle of the right breast.

$$M_{HD} = M_{PM} - M_{GH} \qquad (3\text{-}1)$$

$$M_{VD} = M_{PM} - M_{GV} \qquad (3\text{-}2)$$

Where: M_{PM}, M_{GH}, M_{GV} represent the average mean of PM, GH, and GV local areas;
M_{HD}: the horizontal difference between M_{PM} and M_{GH};
M_{VD}: the vertical difference between M_{PM} and M_{GV}

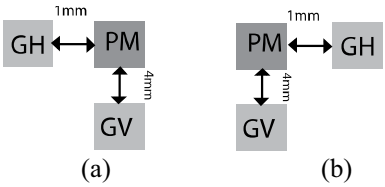


Figure 2: the corner detector

2.2 Detect the cliff of pectoral muscle

Some detection methods were developed to detect the cliff of pectoral muscle. A sigmoid function was used to detect the cliff of pectoral muscle[6]. The technique of edgeflow[14] that detect neighbour with the same direction was adopted to detect the edge of pectoral muscle by Ferrari et al.[5] Those methods were based on the direction of the estimated straight line. To overcome of the superimposition of glandular tissue, an improved MIC algorithm based on the original MIC algorithm[13], was developed to detect the edge of pectoral muscle and it is based on the maximum intensity change with a defined area rather than the direction of the estimated straight line. The improved MIC algorithm searches the maximum intensity change within a defined horizontal range R of a candidate point. The range R is defined as (-1mm, 1mm) of a candidate point, $E_n(x_n, y_n)$. Generally, the texture of the left side area and the right side area of pectoral muscle cliff is different. Two 3x3 pixel matrix was used to detect the difference T_{diff} between the average means of the left side area $MI_l(x, y)$ and the right side local area $MI_r(x, y)$ of a potential edge point. Figure 3 shows the schema of the MIC algorithm. After the process of detecting first point of edge of pectoral muscle, the first point was detected. The process of searching the cliff is based on the detected first point.

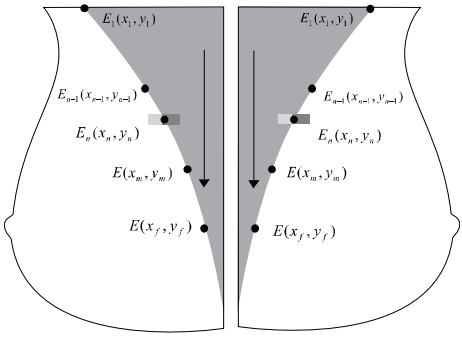


Figure 3: the schema of the MIC algorithm

$$T_{diff} = MI_r(x\ ,y) - MI_l(x,y) \text{ if the } Orientation = left \quad (3\text{-}3)$$

$$T_{diff} = MI_l(x,y) - MI_r(x,y) \text{ if the } Orientation = right \quad (3\text{-}4)$$

Maximum Intensity Change (MIC) Algorithm

1. *Search a candidate point* $E'_n(x'_n, y'_n)$ *with the maximum intensity change within the defined range (-1mm, 1mm) offset of* x'_n
2. *If this candidate point meet the following conditions, it is a true edge point* $E_n(x_n, y_n)$ *along the cliff of pectoral muscle*
   a. $x_{n-1} \le x_n$, *if the* $Orientation = left$
   b. $x_{n-1} \ge x_n$, *if the* $Orientation = right$
3. *Generate a next candidate point* $E_{n+1}(x'_{n+1}, y'_{n+1})$, *where* $x'_{n+1} = x_n$, $y'_{n+1} = y_n + 1$
4. *Repeat the search process until the horizontal corresponding of a candidate point near the boundary of the segmented breast area, or the vertical corresponding near the bottom of segmented breast area.*

2.3 Enclose the pectoral muscle area

The detected edge of pectoral muscle might not enclose the final part of pectoral muscle because of the superimposition of breast glandular tissue. By assuming the undetected edge of pectoral muscle is a straight line, the final point of cliff of pectoral was generated using information of the final detected point $E(x_f, y_f)$ of cliff with the MIC algorithm and the point $E(x_m, y_m)$ located on 80% length of the de-

tected whole cliff. The horizontal corresponding x_m was found by searching the detected points on the cliff with information of y_m.

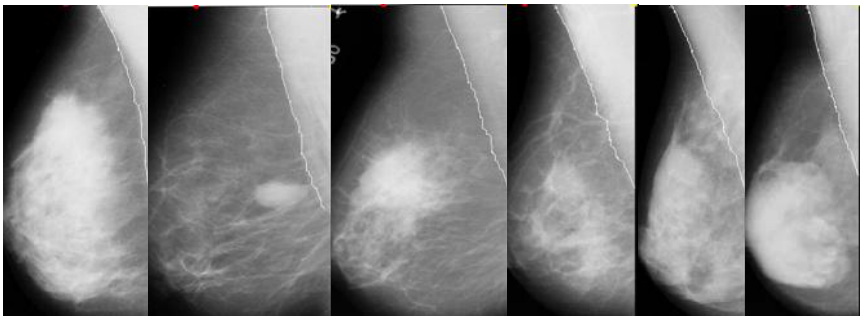
3 Experimentation and Results

The mini-MIAS database[15] was used to validate the proposed approach. Thirty mammographic images were randomly selected from the mimi-MIAS database. In this research, the resolution of the selected mammogram images were transferred to 512x512 pixels at a resolution of 400 μm. The qualitative analysis was carried out to evaluate the results of experiments with three kinds of segmentation quality for pectoral muscle. The results of initial experiment (see Table 1) showed that the proposed detection approach for pectoral muscle can accurately detect the pectoral muscle of 93.3% sample images with high quality, and the results of 6.6% sample images were acceptable. The proposed approach with the corner detection and the improved Maximum Intensity Change (MIC) algorithm is very promising.

Table 1: results of qualitative analysis

categories	Number	Percentage
Excellent	28	93.3%
Acceptable	2	6.6%
Unacceptable	0	0
Total	30	

.



Mdb003 mdb025 mdb081 mdb101 mdb017 mdb037

Figure 4: the detected edge of pectoral muscle

4 Conclusion

The segmentation of pectoral muscle in mammograms is a challenge as the superimposition of breast background tissue and pectoral muscle. Using morphological characteristics of pectoral muscle, a new approach in this research was proposed to overcome the superimposition of breast tissue. The proposed novel approach consists of a corner detector and Maximum Intensity Change (MIC) algorithm. The initial experiment results showed that the proposed approach detected pectoral muscle with high quality. The quantitative measurement evaluation will be carried out in the future.

References

1. Suri, J.S. and R.M. Rangayyan, *Recent Advances in Breast Imaging, Mammography, and Computer-Aided Diagnosis of Breast Cancer*. 2006: SPIE Publications.
2. Yapa, R.D. and K. Harada, *Breast Skin-Line Estimation and Breast Segmentation in Mammograms using Fast-Marching Method.* International Journal of Biological, Biomedical and Medical Sciences, 2008. **3**(1): p. 54-62.
3. Wirth, M., D. Nikitenko, and J. Lyon, *Segmentation of the Breast Region in Mammograms using a Rule-Based Fuzzy Reasoning Algorithm.* ICGST-GVIP, 2005. **5**(2).
4. Karssemeijer, N., *Automated Classification of Parenchymal Patterns in Mammograms.* Physics in Medicine and Biology, 1998. **43**(2): p. 365–378.
5. Ferrari, R.J., et al., *Automatic Identification of the Pectoral Muscle in Mammograms.* IEEE Transactions on Medical Imaging, 2004. **23**(2): p. 232-245.
6. Kwok, S.M., et al., *Automatic pectoral muscle segmentation on mediolateral oblique view mammograms.* IEEE Transactions on Medical Imaging, 2004. **23**(9): p. 1129-1140.
7. Mustra, M., J. Bozek, and M. Grgic, *Breast Border Extraction and Pectoral Muscle Detection Using WAVELET Decomposition.* IEEE, 2009: p. 1428-1435.
8. Raba, D., et al. *Breast Segmentation with Pectoral Muscle Suppression on Digital Mammograms*. in *Proceedings of the 2nd Iberian Conference (IbPRIA 2005)*. 2005. Estoril, Portugal: Springer Berlin / Heidelberg.
9. Ma, F., et al., *Two graph theory based methods for identifying the pectoral muscle in mammograms.* Pattern Recognition, 2007. **40**: p. 2592 – 2602.
10. Zhou, C., et al., *Computerized image analysis: Texture-field orientation method for pectoral muscle identification on MLO-view mammograms.* Medical Physics, 2010. **37**(5): p. 2289-2299.
11. Ferrari, R.J., et al., *Segmentation of Mammograms: Identification of the Skin – air Boundary, Pectoral Muscle, and Fibroglandular Disc*, in *Proceedings of the 5th International Workshop on Digital Mammography*. 2000: Toronto, Canada. p. 573–579.
12. Zhang, Z., J. Lu, and Y.J. Yip, *Automatic Segmentation for Breast Skin-line*, in *Proceeding of the 10th IEEE International Conference on Computer and Information Technology*. 2010, IEEE Computer Society: Bradford, West Yorkshire, UK. p. 1599-1604.
13. Zhang, Z., J. Lu, and Y.J. Yip. *Pectoral Muscle Detection*. in *the 16th International Conference on Automation and Computing (ICAC'10)*. 2010. University of Birmingham, Birmingham, UK.
14. Ma, W.Y. and B.S. Manjunath, *EdgeFlow: a technique for boundary detection and image segmentation.* IEEE Trans. Image Processing, 2000. **9**(8): p. 1375-1388.
15. Suckling, J., et al., *The Mammographic Image Analysis Society Digital Mammogram Database.* Exerpta Medica. International Congress Series, 1994. **1069**: p. 375-378.